FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Navegação de AUVs na proximidade de estruturas usando visão computacional

AUV navigation using computer vision in the proximity of structures

Nuno Miguel Valente Agante

Masters in Electrical and Computer Engineering

Supervisor: Prof. Dr. José Carlos Santos Alves

Second Supervisor: Dr. Andry Maykol Gomes Pinto

July 23, 2018

Resumo

A aquacultura é uma indústria em crescimento que produz quase metade do consumo mundial de peixes. As gaiolas e redes utilizadas nesta indústria necessitam de manutenção frequente. Esta manutenção é feita, atualmente, com processos muito dispendiosos. Sendo necessário recorrer a veículos operados remotamente que necessitam de operadores qualificados e infraestrutura significativa, ou a mergulhadores qualificados, que apresenta limitações significativas devido a preocupações de segurança.

Recentes avanços tecnológicos tornaram possível a substituição de trabalho humano subaquático por AUVs confiáveis e acessíveis (Veículos Submarinos Autónomos). Apesar disso, a inspeção de redes de aquacultura ainda não é significativamente automatizada.

Os processos de localização e navegação dos AUVs são geralmente baseados em sistemas acústicos, que requerem a instalação de uma infraestrutura de apoio e sofrem vários problemas resultantes da conhecida complexidade do canal acústico submarino. Para operações de curto alcance, o recurso à visão computacional é uma maneira viável de melhorar a precisão dos robôs submarinos que operam próximos de estruturas.

Esta dissertação tem como objetivo desenvolver um sistema de posicionamento baseado em visão computacional de camera única para AUVs que operam perto de redes de aquacultura ou gaiolas. O sistema desenvolvido estima os ângulos e a distância em relação a uma rede de aquacultura. Isto é importante para permitir que o veículo autónomo se posicione de forma adequada relativamente à rede, isto, por sua vez, facilita a aquisição de imagens com características adequadas para futuros processos de autommação, assim como a identificação de defeitos.

O sistema de localização relativo calcula os ângulos usando a diferença de áreas causada pela perspectiva. Esta área também é usada para estimar a distância da camera até a rede.

Um sistema foi desenvolvido para produzir imagens sintetizadas que se assemelham a uma situação real, a fim de fornecer um ambiente de teste controlado. Essas imagens foram então usadas para testar e validar o algoritmo de localização.

O sistema foi capaz de determinar os ângulos de *pitch* e *yaw* com um erro menor que 3.5° e a distância com um erro menor que 1.5 vezes o tamanho do lado da malha da rede.

Abstract

Aquaculture is a growing industry that accounts for half of the world's consumption of fish. The cages and nets that are used in this industry need frequent maintenance which is currently done by expensive processes, using remotely operated vehicles which need skilled operators and significant infrastructure, or skilled divers which presents significant limitations due to safety concerns.

Recent technological advancements have made possible the replacement of human labor underwater by reliable and affordable AUVs (Autonomous Underwater Vehicle). Despite this fact, aquaculture net inspection is not yet significantly automated.

Location and navigation processes of AUVs are generally based on acoustic systems, that require the installation of a physical support infrastructure and suffer various problems resulting from the known complexity of the underwater acoustic channel. For close range operation, computer vision is a viable way of improving the precision of underwater robots operating close to structures.

This dissertation aims to develop a single camera vision based positioning system for AUVs operating close to aquaculture nets or cages. The system developed estimates the angles and distance relative to an aquaculture net. This is important since it allows the AUV to position itself in a convenient orientation with respect to the net, which in turn facilitates an improved acquisition of images of the net under inspection for further automation such as identification of defects.

The relative localization system calculates the angles by using the difference in areas caused by perspective. This area is also used to estimate the distance from the camera to the net.

A system was also developed to produce synthesized images which resemble a real situation in order to provide a controlled test environment. These images were then used to test and validate the localization algorithm.

The system was able to determine the pitch and yaw angles with an error of less than 3.5° and the distance with an error of less than 1.5 times the size of the side of the net loops.

Acknowledgments

I would like to acknowledge and thank the following important people who have supported me, not only during the course of this project, but throughout my Masters degree.

Firstly, I would like to express my gratitude to my supervisor Prof. Dr. José Carlos Alves for his unwavering support, guidance and insight throughout this research project.

I would also like to thank Dr. Andry Gomes Pinto for introducing me to the topic as well for the support on the way.

Finally, I must express my very profound gratitude to my parents and to my girlfriend, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Nuno Agante

"No one wants to learn by mistakes,
but we cannot learn enough from successes
to go beyond the state of the art."

Henry Petroski

Contents

List of Figures

List of Tables

List of Abbreviations

AUV	Autonomous Underwater Vehicle
DVL	Doppler Velocity Log
EKF	Extended Kalman Filter
GPS	Global Positioning System
IMR	Investigation Maintenance and Repair
IMU	Inertial Measurement Unit
LBL	Long Baseline
LoG	Laplacian of Gaussian
RANSAC	Random Sample Consensus
ROV	Remotely Operated underwater Vehicle
SLAM	Simultaneous Localization and Mapping
SONAR	Sound Navigation and Ranging
TOF	Time of Flight
USBL	Ultra Short Baseline

Chapter 1

Introduction

In recent years there has been an increase in the usage of machines to automate work. This has made the work of humans easier by improving efficiency and/or reducing danger of those tasks.

When visual feedback is necessary such as for part inspection or guidance when there is a lack of other means of positioning a few tools can be used such as light, ultrasonic sensors or, for more advanced applications, computer vision.

Computer vision in the past few years has become a very powerful inspection tool. With advancements in imaging sensors and computer processing power a larger and more complex number of tasks have been possible to automate.

Aquaculture is an industry that supplies half of the world's demand for fish and other seafood. Due to this demand, there has been a need for increased efficiency and sustainability. However, automation has not yet been widely implemented in this industry.

Recently, underwater vehicles have become reliable and affordable platforms for performing various underwater tasks. With the increase in processing power and greater energy efficiency, the autonomy of these vehicles has also increased. Nowadays, an affordable Autonomous Underwater Vehicle (AUV) is capable of performing tasks that previously required a human operator.

In order to navigate underwater, a AUV needs to be able to know its position. Because radio waves have a very limited range in water, radio-based positioning systems, such as the GPS, are not viable. Sound, however, has better performance and because of that, acoustic systems are widely used for underwater positioning. The biggest drawback is the necessity of additional infrastructure that needs to be installed prior to the deployment of vehicles.

1.1 Motivation

One of the tasks that can be automated with help from computer vision and AUVs is the inspection of the nets or cages used in aquaculture. This is an important task which needs to be performed frequently in order to ensure the proper functioning of the aquaculture farm, since a small hole can lead to the loss of the whole fish population.

Since the farms are not usually rigidly fixed to the sea bottom, they are subject to movement due to waves and tides. For this reason, it is more important to know the position relative to the cage or net being inspected as opposed to the absolute position.

So, this thesis focuses on the positioning of a AUV in relation to a uniform structure such as a net or cage used in aquaculture. This involves the determination of the vehicles' pose or orientation, and the distance from the vehicle to the net. The depth can easily be obtained by other means, such as pressure sensors. The horizontal movement can be estimated using other image processing techniques based on optical flow.

1.2 Goals

The main goal of this thesis is the development of a system that is able to transform the video feed of a AUV into data that permits the navigation relative to a uniform structure.

In order to develop such a system it is necessary:

First, the study of the requisites for the application of a single camera system in a AUV/ROV for the inspection of aquaculture cages or nets.

Secondly, the collection of visual data from a virtual environment and a controlled physical environment.

Then, the implementation of a system for net identification and pose estimation.

Lastly, the validation of this method through testing in simulated and real environments.

1.3 Document Structure

After this introduction, this document is organized in 5 more chapters. Chapter 2 is the state of the art, there is presented the work previously done in the field of underwater vehicle localization as well as a brief explanation of the main image processing methods.

In Chapter 3 the approach to the problem is described together with tests to evaluate this approach.

The implementation of the approach is described in Chapter 4, where the algorithm described in the previous chapter is implemented and the work done is described.

Next, in Chapter 5, the results of the implementation are shown together with a description of the error correction methods used in order to improve the results.

Finally, in Chapter 6, an evaluation of the results of this work is given as well as possible future work.

Chapter 2

State of the Art

This chapter explores the techniques used for underwater navigation, more specifically the methods used for underwater localization. The main focus of this work is vision based localization so image acquisition and processing methods are researched as well. Finally, an overview of the current aquaculture automation status is given.

2.1 Navigation

Navigation is a field of study that focuses on the movement from one place to another. In the past few decades navigation of autonomous vehicles has gained a lot of interest [26]. *Leonard and Durrant-Whyte (1991)* [22] describe the problematics of navigation using three simple questions:

- *'Where am I?'* This is related to the current position of the subject. It is the information that is able to be acquired from the environment that allows the subject to know the position in relation to its surroundings.
- *'Where am I going?'* This relates to the objective or destination. It is the purpose of the navigation.
- *'How do I get there?'* This describes the plan that needs to be made in order to achieve said goal given the location of the subject.

More recently, the problem has also been divided into three blocks: localization, path planning, and obstacle avoidance [20] [18]. This division is more accurate since more recent applications have given less emphasis on the final destination of the vehicle and more importance has been given to the path taken. Despite this shift of focus, accurate localization is still one of the main components of a proper functioning system.

2.2 Localization

Knowing the accurate location of the system is fundamental in mobile robot applications. Because of that there has been significant research in the field and great number of systems, sensors, and

techniques have been developed. It is still an active field of study and as such there are more solutions being made [32].

Since there is no single simple solution to the problem *Borenstein et al. (1996)* [7] divided the existing partial solutions in two groups which include seven distinct categories:

1. Relative Position Measurements (also called Dead Reckoning)
   (a) Odometry
   (b) Inertial Navigation
2. Absolute Position Measurements (Reference-based systems)
   (a) Magnetic Compasses
   (b) Active Beacons
   (c) Global Positioning Systems
   (d) Landmark Navigation
   (e) Model Matching

The first group contains methods that track changes in the system and depend on data from the previous instant to know their current state. The second group has the methods which do not need the previous data to estimate their location. Most solutions have a combination of methods from both groups.

2.3 Underwater Localization

Most radio-frequency based positioning systems used in land or air cannot be used underwater since radio waves are strongly attenuated underwater, especially in sea water [12].

For AUVs, solutions can be divided into three main categories according to Figure 2.1:

- Inertial/Dead Reckoning: as mentioned in Section 2.2 it relates to the measurement of change in the system and calculation of current state based on previous measurements.
- Acoustic Transponders and Modems: this involves the usage of beacons and calculation of time-of-flight for an acoustic signal to obtain the position in relation to the beacons.
- Geophysical: this type of positioning is based on external information obtained from the environment. It is necessary for the system to have the capability to acquire and process data about its environment.

The complete positioning system is usually composed of a few of these elements, all the data is then processed and filtered and an estimation is calculated [2] [36].

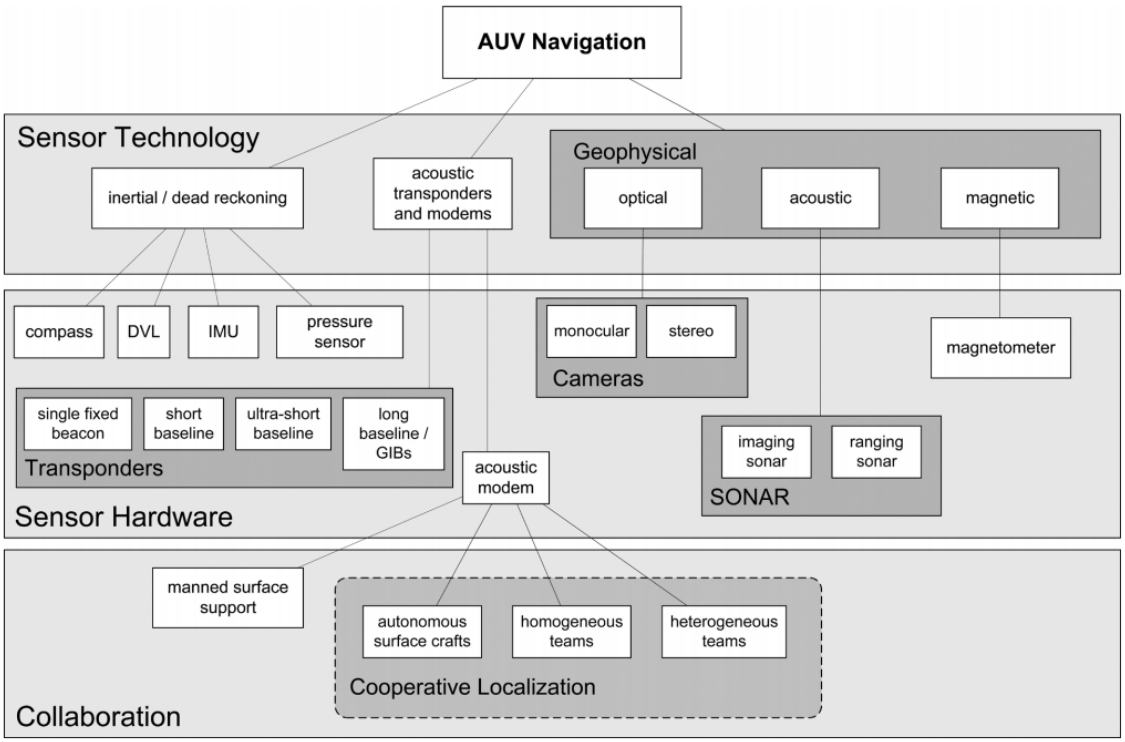


Figure 2.1: Outline of underwater navigation classifications. Adapted from [25]

2.3.1 Dead Reckoning and Inertial Navigation

Dead Reckoning is a term used in navigation for hundreds of years, as *Bowditch, N.* defines it:

"Dead reckoning allows a navigator to determine his present position by projecting his past courses steered and speeds over ground from a known past position." [8].

In an AUV context, it refers to systems that use the previous position data together with information about changes in the vehicle to calculate the current position.

The sensor data used for this calculation can be obtained from sensors that provide the vehicle's positional rate of change or its heading. Some of these sensors, as seen in Figure 2.1, are:

- Compass, provides the heading in relation to the earth's magnetic field.
- Doppler Velocity Log (DVL), calculates a velocity vector using acoustic sensors and provides a three dimensional vector of speed in relation to the surrounding water.
- Pressure Sensor, gives the depth based on pressure.
- Inertial Measurement Unit (IMU), consists of a combination of gyroscopes, accelerometer and sometimes magnetometer to estimate the angular velocity and acceleration of the vehicle.
- Altimeter, this is a low resolution Sound Navigation and Ranging (SONAR) system which provides the distance to the seabed.

Inertial navigation systems are a type of dead reckoning system that use an IMU or a set of sensors that provide the rate of change in heading and velocity [5]. Because of this the data needs to be integrated, once to obtain velocity and heading measurements and a second time for positioning. Due to the integration, small errors in sensor data can cause unbounded errors in the position estimation.

Systems have been developed to improve accuracy in the estimation [37], however the usual approach is to use other forms of localization together with dead reckoning and apply a filter that allows the readjustment of the position estimate [1].

2.3.2 Acoustic Navigation

Underwater, sound has a higher speed and range compared to air [19], but also significant drawbacks [25]:

- Small bandwidth, which means communicating vessels often need the same frequency to communicate and frequency sharing algorithms are necessary.
- Low data rate, which generally constrains the amount of data that can be transmitted.
- High latency since the speed of sound in water is around 1500 m/s which is slow compared with the speed of light.
- Variable sound speed due to changes in water temperature, salinity, and pressure.
- Multi-path transmissions, i.e. sound reflects when it reaches either the top or bottom surface which results in multiple paths to the same destination.
- Unreliability, this implies the need for the system to handle data loss in communications.

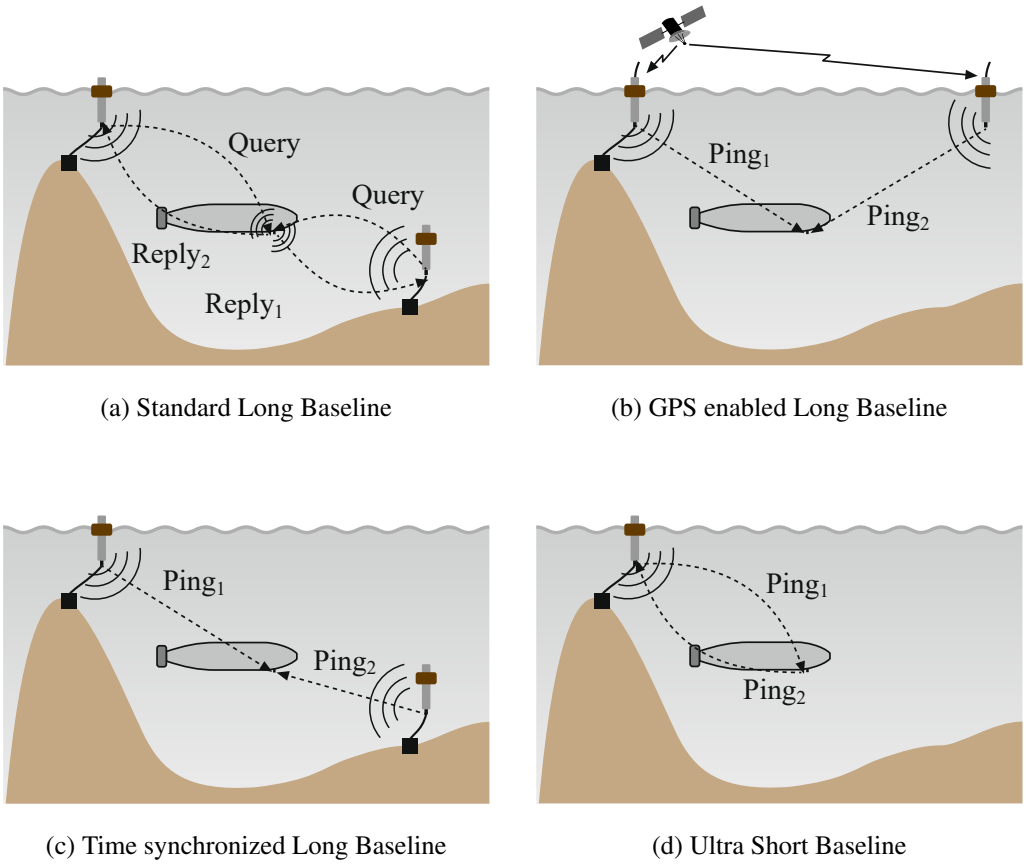
In acoustic based navigation, location is calculated using sound waves and knowledge of its speed underwater. The time-of-flight (ToF) is the time it takes for a sound wave to travel a certain distance underwater.

The most common form of acoustic navigation takes advantage of beacons positioned in specific locations. Most systems can be grouped in two categories: Long Baseline (LBL), and Ultra Short Baseline (USBL) [23].

2.3.2.1 Long Baseline

Standard LBL These systems consist of two or more beacons positioned either at the surface or a short distance from the seabed, the latter is used mostly for deep sea missions (Figure 2.2a).

A request is sent from the AUV and the beacons respond at a set unique frequency and with a specific delay. The different frequency allows the AUV to identify which beacon has sent the response and the delay ensures that the beacons do not interfere with each other. This allows only a single vehicle to make a request at a time, which decreases the frequency of requests that can be made as the number of vessels increases.



(a) Standard Long Baseline

(b) GPS enabled Long Baseline

(c) Time synchronized Long Baseline

(d) Ultra Short Baseline

Figure 2.2: Acoustic navigation examples. Adapted from [23]

The delay between the request and the answer signal allows the vehicle to calculate its distance from the beacon. Then the AUV can trilaterate its position using distance information obtained from a few beacons and data stored in the vehicle about their locations.

The error and range of these systems varies with the frequencies used. The higher frequencies allow for a more precise location but have restricted range; the lower frequencies have a longer range but the error is also increased.

These systems are vulnerable to outliers resultant from multi-path propagation as mentioned in Section 2.3.

LBL Variations To attenuate some flaws in standard LBL namely the fact that only a vehicle can query the system at a time, some variations have been developed. One of the solutions involves the usage of synchronized clocks, which removes the necessity for a request originating from the AUV by periodically sending a signal from the beacon. The vessel can use the stored information about the time at which the signal was sent and the time of arrival to calculate the distance from the beacon.

Other improvements include the use of GPS enabled beacons and transmission of accurate position in the signal sent. This makes it possible for the beacons to not be anchored and removes

the necessity to store the beacon position before deployment.

2.3.2.2 Ultra Short Baseline

USBL varies from long baseline systems in the way it handles the signal sent from the beacon. Instead of a single receptor it uses multiple receiving elements located very close to one another. This allows the vehicle to compute its bearing using the phase shift from the different receiving elements. The distance can be calculated the same way as in LBL systems, by measuring the time between the query and the answer.

This system has the advantage of needing only one beacon to perform the computation of its localization.

2.3.2.3 Cooperative Navigation

Some applications require a large area to be surveyed or a higher redundancy of data, for these applications it can be useful to use a fleet of AUVs. However, as mentioned in Section 2.3.2.1, an increased number of vehicles decreases the frequency at which the positioning requests can be made.

Cooperative navigation is based on the sharing of information between vehicles on a fleet of AUVs. Since a vehicle can have more accurate positioning data than others in the same group, by relaying information of the details of its estimate including errors, it can improve the localization estimation of the whole fleet. [4] [11].

2.3.3 Geophysical Navigation

The installation of beacons can, for some applications, be impractical. One approach that can be taken is to use geophysical data. This can be of various types like bathymetric, magnetic field, or gravitational field. Assuming that there is enough variation and the information is mapped in enough detail it is possible to determine the position of the vehicle. This is akin to human practices, as maps have been used as a mean of localization for millennia; we use landmarks or features of the landscape to figure out our position.

In an AUV context it presents other difficulties since obtaining the map in enough detail can be very expensive, and matching sensor data to the map can be computationally complex.

Sometimes it is not feasible to have a map of the location beforehand, such as when the mission is mapping the location, in most of those situations a technique called Simultaneous Localization and Mapping (SLAM) is used [21].

2.3.3.1 Simultaneous Localization and Mapping

In terrestrial mobile robotics there has been significant research in navigation systems that do not require a priori knowledge of the environment and have bounded error growth. The aim of SLAM is to identify and map landmarks to restrict the error in odometric data.

The major problem consists in the fact that to construct a precise map it is necessary an accurate trajectory estimate, and to bound the odometry data it is necessary a precise map. An illustration of this problem can be seen in Figure 2.3.

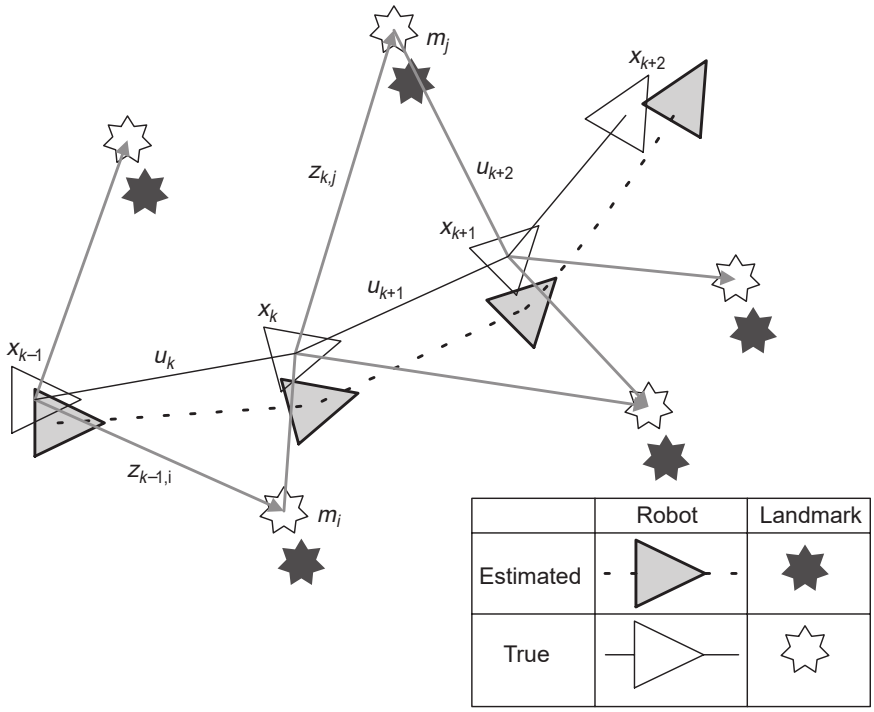


Figure 2.3: The essential SLAM problem: the true locations are never known. Adapted from [13]

The problem of SLAM has many difficult issues, among which are: how to efficiently map large scale environments, correctly associate measurements, and robustly estimate map and vehicle trajectory [33].

In sub-aquatic environments SLAM is mostly used with bathymetric sensors such as bottom facing sonar or optical sensors, sonar provides a range between 10m and 100m, while optical sensors provide a practical range up to 10m [21]. It also encounters other problems such as the identification of features since naturally occurring features cannot be easily identified by a single point feature as it is required by most current SLAM approaches [30].

2.4 Underwater Computer Vision

Optical sensors have a limited range underwater due to water turbidity and poor lighting conditions, they have however the advantage when the vehicle is positioned in close range to the objects of interest.

Because of the characteristics of natural underwater features and the range available to underwater optical sensors, most vision based approaches take advantage of man made structures. These usually possess more spacial variation which is useful for visual tracking algorithms. Man made

structures also allow for some a priori knowledge of the size, shape, or other physical properties of the subject. A few examples are surveying of submerged archaeological sites [17], shipwreck photography [6], pipeline or cable tracking [31] [24].

Another alternative is to use underwater vision to track markers made specifically for the purpose [14], this approach varies from the others in the fact that it is a relative positioning approach.

Apart from the specific characteristics of underwater vision, such as low range and noise introduced by light dispersion in the water, the image processing element is not different from its land based counterpart.

The overall objective of computer vision is to extract enough information from images obtained from a scene in order to be able to interact with it. The usual steps taken are as shown in Figure 2.4: acquisition, pre-processing, segmentation, feature extraction, and feature classification.

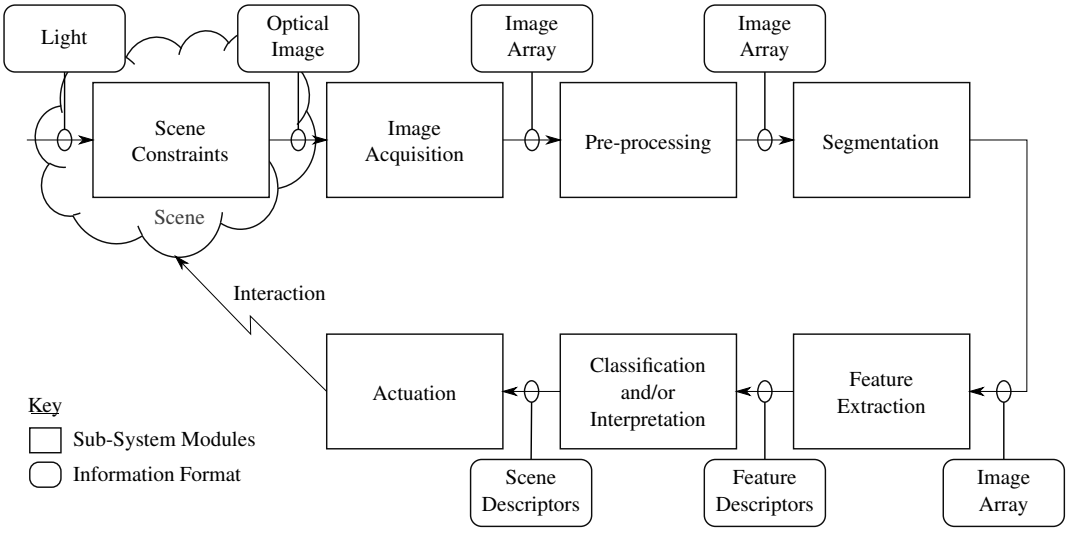


Figure 2.4: Overview of a Modular Machine Vision System. Adapted from [3]

2.4.1 Image Acquisition

Before any processing is done it is necessary to obtain an image. A digital camera transforms real world three dimensional data into a two dimensional image. The finite projective camera matrix is the representation of this transformation. It can be seen in (2.1) where $[u\ v\ 1]^T$ are the coordinates of the pixel in the image and $[x_w\ y_w\ z_w\ 1]^T$ are the world coordinates of the point it represents.

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{2.1}$$

The camera matrix can be decomposed into two matrices and a point:

$$P = KR[I|-\widetilde{C}], \tag{2.2}$$

where $\widetilde{C}$ is the camera center in relation to the world coordinates, K is the matrix of intrinsic camera parameters (2.3), and R is the camera rotation matrix.

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.3}$$

where:

- α_x is the scale factor in the x-coordinate direction,
- α_y is the scale factor in the y-coordinate direction,
- s is the skew, in a typical situation $s = 0$.
- $(x_0, y_0)^T$ are the coordinates of the camera center in the image coordinate system.

The estimation of these parameters is done through a calibration process which usually involves taking pictures of a test pattern in different positions [16].

2.4.2 Image Pre-Processing

Image pre-processing is done at the pixel level, and is usually the first step in the image processing chain. The input and output of this process is an image. The principal aim is to suppress undesired distortions and enhance features that will be important in further processing. It can be classified into three categories with respect to the number of pixels taken into consideration: pixel brightness, geometric, and local neighborhood based transformation [29].

2.4.2.1 Pixel Brightness Transformations

Pixel brightness transformations as the name suggests alter the brightness of the pixel. They can be of two types, brightness correction and gray-scale transformations.

Brightness Correction This is useful when the image presents systematic changes in pixel brightness, e.g. vignetting. The error can be calculated by using a test image of known brightness and comparing it with the obtained image. This transformation represents the changes in brightness in relation to the pixel position in the image.

Gray Scale Transformations These are not dependent in the position of the pixel and are applied to every pixel following the same criteria. The most common transformations are represented

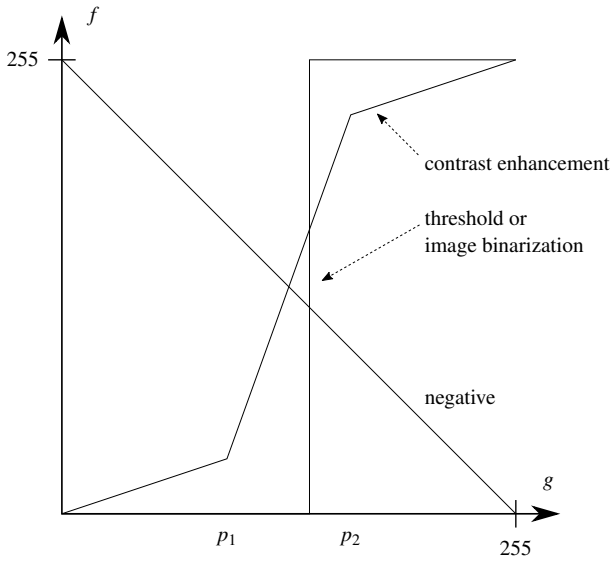


Figure 2.5: Common gray-scale transformations

in Figure 2.5 and are as pictured: thresholding, contrast enhancement, and the negative transformation.

Thresholding is a form of binarization which transforms every pixel according to a determined value: the values that are higher become the maximum possible, the lower values are transformed into the lowest possible.

The negative transformation simply inverts the brightness of the pixel.

Finally, contrast enhancement is a transformation most useful for human viewing. The aim is to improve contrast by making every brightness have the same quantity of pixels, this is not completely possible with a digital image since the brightness values are integers. What happens in reality is a shift of the values of brightness and as such some brightness values are not represented in the final image.

2.4.2.2 Geometric Transformations

These transform the image through defined geometric transformations. It involves two steps, pixel relocation and brightness interpolation.

Pixel Relocation This is a coordinate change, it can be represented by a vector function T that maps the original image pixels to a new value:

$$x' = T_x(x,y), \text{ and } y' = T_y(x,y)$$

Brightness Interpolation The previous transformation does not usually fit the discrete grid of the image. So, to find the intensity value for each pixel, some approximations are necessary. The most common interpolation algorithms are: nearest-neighbor, linear, and bi-cubic.

Nearest-neighbor does a simple rounding operation, the pixel is assigned the value which is closest. The biggest problem is the loss of straightness of some lines which might appear step-like after the transformation as can be seen in Figure 2.6a.

Linear interpolation takes into account the four closest neighbors and does a weighted average where the weight is the distance from the pixel, so the final intensity might not be the same as the original image. It has a smaller impact in the straightness of lines but might cause some blurring due to the averaging element as can be seen in Figure 2.6b.

Bi-cubic has a more complex approach, it makes a weighted average of the 16 closest pixels where the weight follows a bi-cubic polynomial surface. The final image does not suffer as much from the blurring that occurs in linear interpolation neither from the line distortion that occurs from nearest neighbor as can be seen in Figure 2.6c.

(a) Nearest neighbor (b) Linear (c) Bi-cubic

Figure 2.6: Examples of brightness interpolation, original image was scaled up by a factor of 3

2.4.2.3 Local Neighborhood Transformations

Local neighborhood pre-processing can serve two purposes, reducing noise or small artifacts, and enhancing edges by accentuating changes in brightness in the image. These two operations can be considered opposites since noise reduction usually has the side effect of smoothing edges which the second operation tries to enhance; the opposite is also true as edge enhancement will accentuate noise present in the image because noise is presented as sharp brightness changes [38].

Smoothing Image smoothing is a process that aims to eliminate noise from an image. One approach is to do a weighted average of the neighboring pixels. This average can have a uniform weight or more commonly a higher weight for the center pixels and a lower one for the edges. These filters can be applied by doing a discrete convolution with a mask (2.4), h_1 for the uniformly weighted average and h_2 and h_3 for examples of the second type, for the case of a 3×3 neighborhood.

$$h_1 = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}, \qquad h_2 = \frac{1}{10}\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{bmatrix}, \qquad h_3 = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \tag{2.4}$$

A Gaussian blur is another common type of smoothing, this uses the normal Gaussian distribution to generate the kernel of the transformation. Its kernel can be obtained using the formula:

$$G(x,y) = e^{-\frac{(x^2+y^2)}{2\sigma^2}}, \tag{2.5}$$

where x and y are the distance from the origin and σ is the standard deviation of the associated probability distribution. The size of the kernel will be proportional to σ.

Another approach is rank filtering, where the pixel neighborhood is ordered into a sequence, then the output intensity is selected from the sequence according to its position. A common application of this type of filter is median filtering where the median is chosen, i.e. the middle value from the sequence. This is useful to remove sparse noise, also called salt and pepper noise.

Edge Enhancement An edge is an abrupt change in image brightness. It is a property attached to the pixel and can be calculated by evaluating the properties of its neighborhood. This is done usually by approximating the derivative of gradient by convolving it with a mask that enhances differences in the image. Common operators are Sobel, Prewitt, and Laplace.

Sobel (2.6) and Prewitt (2.7) operators both use a set of 8 masks for a 3×3 neighborhood, these 8 masks can be obtained from the first by simple rotation. They approximate the first derivative and the convolution with each mask returns an image where the highest values mark the highest differences in each corresponding direction in the original image. Sobel is used mostly to detect horizontality and verticality of the edges, in that purpose only two of the masks are used h_1 and h_3.

$$h_1 = \begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}, \quad h_2 = \begin{bmatrix}0 & 1 & 2\\ -1 & 0 & 1\\ -2 & -1 & 0\end{bmatrix}, \quad h_3 = \begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}, \ldots \tag{2.6}$$

$$h_1 = \begin{bmatrix}1 & 1 & 1\\ 0 & 0 & 0\\ -1 & -1 & -1\end{bmatrix}, \quad h_2 = \begin{bmatrix}0 & 1 & 1\\ -1 & 0 & 1\\ -1 & -1 & 0\end{bmatrix}, \quad h_3 = \begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix}, \ldots \tag{2.7}$$

The Laplace operator is different, it approximates the second derivative instead and gives the edge magnitude only. The approximation usually involves the convolution with a 3×3 mask, the most basic ones can have either a neighborhood of 4 or 8:

$$h_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad h_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Some variations exist which give different weights to the central pixel or to its neighborhood.

Edge Detection Edge detection can be done by finding maxima in the results of the first derivative approximation. This approach has the disadvantage of being sensitive to noise and dependent on the size of the edge.

Another approach is to use zero crossings in the second derivative, e.g. Laplace operator. This is usually done in conjunction with a smoothing filter to reduce the influence of noise. A common operator is the Laplacian of Gaussian (LoG) which is a combination of both Gaussian smoothing and the Laplace operator.

To reduce error and improve detection of edges multiple algorithms have been developed, one of the most well known is the Canny edge detector [10], it is an algorithm that can be summarized in seven steps:

1. Convolve an image with a Gaussian of scale σ (2.5).
2. Estimate local edge normal directions for each pixel in the image.
3. Find the location of the edges using non-maximal suppression.
4. Compute the magnitude of the edge.
5. Threshold edges in the image with hysteresis to eliminate spurious responses.
6. Repeat steps (1) through (5) for ascending values of the standard deviation σ.
7. Aggregate the final information about edges at multiple scale using the 'feature synthesis' approach

From these, usually the last two steps are omitted and only the first five are performed.

2.4.3 Morphology Operations

After the binarization of an image, some unwanted noise might still be present. In order to remove some of those imperfections, morphology operations can be used.

Morphology operations consist in a non-linear operation that is based on the positioning of the pixels in the image and not in their numerical values [28].

2.4.3.1 Basic Operators

The basic operators of image morphology consist in the association of a small binary shape with an image. This shape is called a structuring element and it is used to probe the neighborhood of every pixel in the image.

Dilation and Erosion Dilation of a binary image produces an image where ones represent all the pixels which the structuring element intersected with a pixel in its neighborhood. This gradually increases the area of the regions and bridges small gaps.

Erosion Produces an image where ones represent every pixel where its neighborhood matches the structuring element. This one increases holes and gaps, and removes small details.

Opening and Closing Opening consists in the sequence of an erosion followed by a dilation. The dilation separates regions that are connected only by a thin line of pixels. The dilation then restores the size of the regions that remain.

Closing is the reverse sequence i.e. dilation followed by erosion. The dilation bridges gaps and fills holes, and the following erosion restores the regions' original size.

2.4.4 3D Reconstruction from 2D Images

There is no unified theory of 3D vision as each application can have its own interpretation. The basic objective is an addition of an extra dimension, either using movement, multiple cameras or even a pre existing model of the object. What most applications have in common is the necessity of associating images with each other or with a pre existing assumption of the scene.

In order to match a two dimension representation of a scene with another something called features is used. These can be of several types and their objective is to be able to represent the image using only its most important points. Then it is necessary to find a correspondence between these features and their pair [16].

2.4.4.1 Feature Detection and Matching

Feature detection algorithms can be of three types, corners or points of interest, blobs, and region based. The feature detection method can have three main properties which are rotation, scale, and affine invariance. Rotation and scale invariant are features that can be detected regardless of the image rotation. Scale invariant, is a feature resistant to scale changes in the image. Affine invariant is a feature that can be detected even if the image has suffered an affine transformation.

The quality of feature detection methods can be evaluated using four criteria: repeatability, location accuracy, robustness, and efficiency. Repeatability measures the percentage of features detected in two different images of the same subject. Location accuracy measures how close in the real object are the matches. Robustness is how resistant the feature is to deformations in the object. Efficiency is the amount of processing necessary for the feature identification and matching [34].

Table 2.1: Summary of feature detection and matching algorithms. Adapted from [34]

	Type			Invariance			Properties			
	Corner	Blob	Region	Rotation	Scale	Affine	Repeatability	Accuracy	Robustness	Efficiency
Harris	✓	–	–	✓	–	–	High	High	High	Medium
Hessian	–	✓	–	✓	–	–	Medium	Medium	Medium	Low
SUSAN	✓	–	–	✓	–	–	Medium	Medium	Medium	High
Harris-Laplace	✓	✓	–	✓	✓	–	High	High	Medium	Low
Hessian Laplace	✓	✓	–	✓	✓	–	High	High	High	Low
DoG	✓	✓	–	✓	✓	–	Medium	Medium	Medium	Medium
SURF	✓	✓	–	✓	✓	–	Medium	Medium	Medium	High
Harris-Affine	✓	✓	–	✓	✓	✓	High	High	Medium	Medium
Hessian-Affine	✓	✓	–	✓	✓	✓	High	High	High	Medium
Salient Regions	✓	✓	–	✓	✓	✓	Low	Low	Medium	Low
Edge-based	✓	–	–	✓	✓	✓	High	High	Low	Low
MSER	–	–	✓	✓	✓	✓	High	High	Medium	High
Intensity-based	–	–	✓	✓	✓	✓	Medium	Medium	Medium	Medium
Superpixels	–	–	✓	✓	✓	✓	Low	Low	Low	Low

2.5 Aquaculture Automation

The growth of aquaculture has led to a need for greater efficiency and the need to expand to new areas which are further away from coast. These new areas are harder to maintain and significantly more dangerous so solutions which decrease human intervention are necessary [35].

2.5.1 Cage Maintenance

Investigation Maintenance and Repair (IMR) are frequent operations in an aquaculture operation. The frequency at which they occur is controlled by the company in charge of the farm. It needs nevertheless to be frequent enough that the fish have a good growing environment.

Currently, aquaculture cage maintenance is done through either specialized divers or Remotely Operated Vehicles (ROV). The use of specialized divers has significant costs so routine IMR is mostly done using ROVs.

Generally, current industrial ROVs need to be manually operated by specialized personnel and require support of expensive machinery for deployment, operation and recovery. Most current systems do not possess any automatic control functions or autonomy. Because of that, the efficiency of operations is dependent on the experience of the ROV operator [27].

In a ROV there is no necessity for active pitch and roll control since it is possible to design a vehicle which is stable in those directions [15].

Chapter 3

Algorithm

The objective of the algorithm is to determine the orientation of the vehicle relative to the plane that approximates the net surface. It should also be able to calculate the distance between the center of the camera and the net surface.

The basic scenario considered is an AUV with a camera pointing in front of it. The coordinate axis which represent both the camera and the vehicle have the same origin.

The range of pitch and yaw considered was $[-80^\circ, 80^\circ]$. The reason for this will be further explained.

It was also assumed that the AUV navigates sufficiently close to the net that the camera is able to capture an image with sufficient detail to recognize the loops of the net, and enough contrast to be able to separate the background from the net.

3.1 Coordinate System

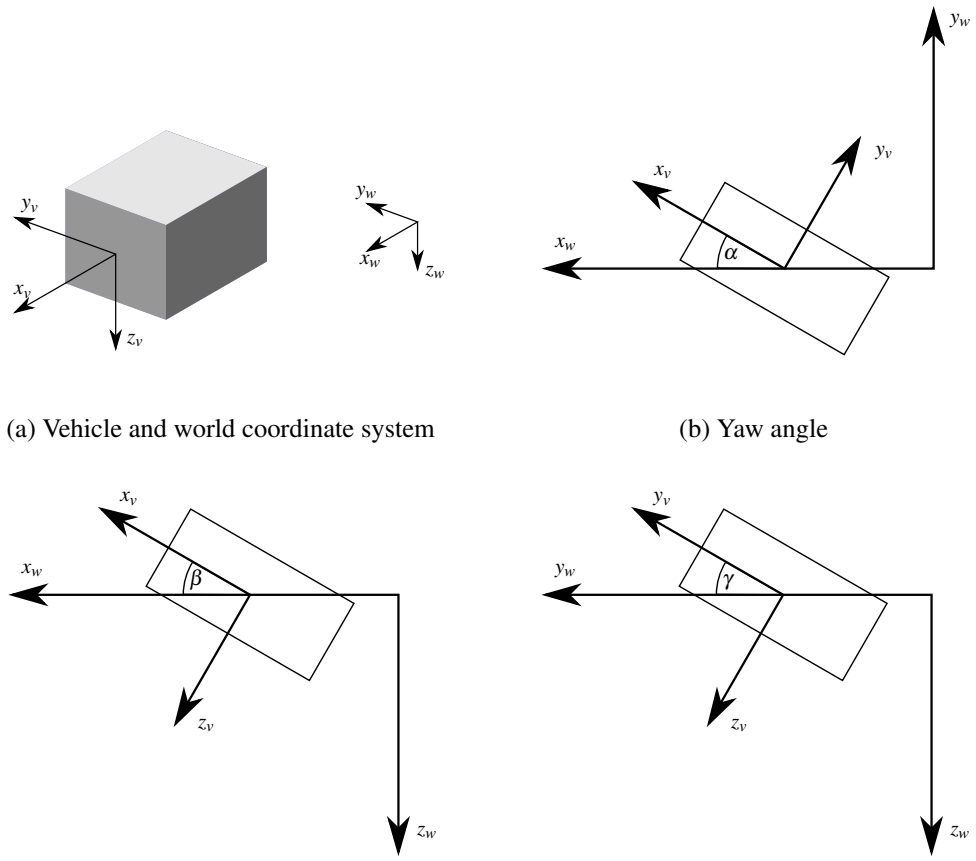
The coordinate conventions for each system are different so it is important to define the system used. The vehicle coordinate convention can be seen in Figure 3.1 where x_w, y_w, and z_w are axis of the world coordinate system and x_v, y_v, and z_v are the axis of the vehicle coordinate system.

The camera coordinates can be seen in Figure 3.2, here, since the camera is positioned in the front of the vehicle, the camera's x_c, y_c, and z_c axis coincide with the vehicle's y_v, z_v, and x_v axis, respectively. For simplicity, only the camera coordinate system was used and the translation into vehicle coordinates was only done at a final stage.

It can also be noted that the u and v axis represent the pixel coordinates, and x and y are the image coordinates.

3.2 Input Data Generation

Initially, the calculations were done for a square of side l with a center at $(0,0,0)$ and coplanar with the plane defined by the x and y axis.



(a) Vehicle and world coordinate system

(b) Yaw angle

(c) Pitch angle

(d) Roll angle

Figure 3.1: Coordinate system and angles of the vehicle. Adapted from [15]

After, a grid of 10×10 points was used as input data, this grid had a center at $(0,0,0)$ and was coplanar with the plane defined by x and y axis.

Both of these data sets were generated in three dimensions and then rotated using the Tait-Bryan angles relative to the vehicle and translated in the directions desired.

A representation of the data obtained by the camera was calculated using the perspective matrix.

The Tait-Bryan angles are widely in nautical applications and are called *yaw*, *pitch*, and *roll*. These represent three rotations around the stationary world axis (extrinsic rotations) or around the axis of the vehicle which changes after each rotation (intrinsic rotations).

In this case, the axis of the camera remained stationary so the rotations were applied to the points in the net. The conventional rotation order is $x-y-z$ extrinsic which means a rotation around the x_w, then y_w and finally z_w. This was translated into an extrinsic $z-x-y$ rotation around the camera axis. The rotation of the points in the net was the same as a rotation of the

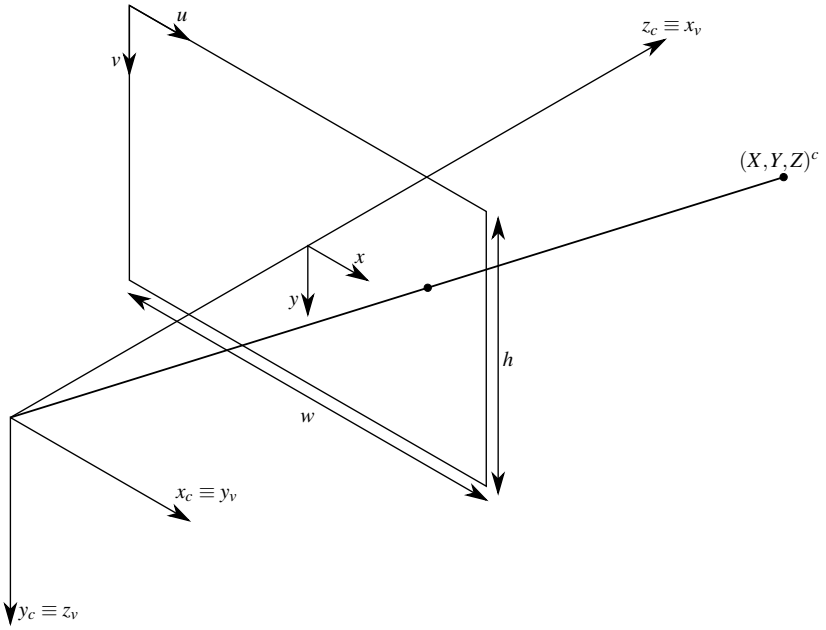


Figure 3.2: Camera Coordinates

camera with the inverse angle for the rotations around y_c and z_c.

$$\begin{aligned}\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ p'_\omega \end{bmatrix} &= T\,R_{x,v}R_{y,v}R_{z,v}\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \\ &= T\,R_{z,c}R_{x,c}R_{y,c}\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix},\end{aligned} \tag{3.1}$$

$$\begin{aligned} T &= \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad R_{y,c} = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ R_{x,c} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\beta) & \sin(\beta) & 0 \\ 0 & -\sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R_{z,c} = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{3.2}$$

where α, β, and γ are the yaw, pitch, and roll of the vehicle, respectively. These rotations were done with respect to the camera axis and then translated in the z_c axis.

The projective matrix used was, from (2.2):

$$P = \begin{bmatrix} \frac{1}{\tan\left(\frac{\theta}{2}\right)} & 0 & 0 & 0 \\ 0 & \frac{\rho}{\tan\left(\frac{\theta}{2}\right)} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{3.3}$$

where ρ is the output video aspect ratio and θ is the field of view of the theoretical camera. It can be noted that the skew parameter $s = 0$ since this is the most common case in commercial cameras.

Then, to get the normalized coordinates of the grid, the x, y, and z components of the points' coordinates were divided by the ω component.

Since the last rotation is around the z_c axis, it represents an affine rotation of the image which will not have influence on the area of the polygon and can be removed for simplicity. Each point normalized 2D representation is then given by:

$$\begin{aligned} p'_x &= -\frac{v_x + x\cos(\alpha) + z\sin(\alpha)}{\tan\left(\frac{\theta}{2}\right)\left(v_z - y\sin(\beta) + z\cos(\beta)\cos(\alpha) - x\cos(\beta)\sin(\alpha)\right)}, \\ p'_y &= -\frac{v_y + y\cos(\beta) + z\cos(\alpha)\sin(\beta) - x\sin(\beta)\sin(\alpha)}{\tan\left(\frac{\theta}{2}\right)\left(v_z - y\sin(\beta) + z\cos(\beta)\cos(\alpha) - x\cos(\beta)\sin(\alpha)\right)}, \end{aligned} \tag{3.4}$$

which is then multiplied by the window height and added the coordinates of the window center to obtain the pixel representation of the point.

3.3 Distance from Area

With the a priori knowledge of the net size and real area of the net loops distance was calculated using the relation between the area of the 2D representation of the net and the real area.

The area of a 2D polygon representation of a 3D square with side l at a distance v_z and with $v_x = v_y = 0$ is given by:

$$\mathrm{A} = \frac{16\,l^2\,v_z^2\,|\cos(\alpha)\cos(\beta)|}{\tan\left(\frac{\theta}{2}\right)^2 |2v_z + \sigma_1 + \sigma_2|\,|2v_z - \sigma_1 + \sigma_2|\,|2v_z + \sigma_1 - \sigma_2|\,|2v_z - \sigma_1 - \sigma_2|}, \tag{3.5}$$

where

$$\begin{aligned} \sigma_1 &= l\cos(\alpha)\sin(\beta) \\ \sigma_2 &= l\sin(\alpha) \end{aligned}$$

Since the objective is to estimate the angle, the approximation made was to remove the angle component. This will add an error in the distance estimation which is related to the angle of the

net. However, since this error will be shared by all the net loops, this approximation allows for an estimation of the angle which can then be used for error correction.

From this final approximation the distance can be estimated using the formula:

$$v_z \approx \frac{l}{\tan\left(\frac{\theta}{2}\right)\sqrt{A}} \tag{3.6}$$

3.4 Yaw and Pitch Estimation

After obtaining a distance estimation, the next step is to obtain an angle from the plane formed by the net. This can be achieved by extracting the centroid from each polygon and, using the approximation for the distance, calculate the 3D position of the centers of the loops in the net.

From a set of three non-collinear points it is possible to calculate a plane that represents the net. From this plane the pitch and yaw can be calculated through simple trigonometry as can be seen from Figure 3.3.

$$\begin{gathered} A=(a_x,a_y,a_z), \quad B=(b_x,b_y,b_z), \quad C=(c_x,c_y,c_z), \\ \overrightarrow{AB}=B-A, \quad \overrightarrow{AC}=C-A, \\ \vec{n}=\overrightarrow{AB}\times\overrightarrow{AC}, \end{gathered} \tag{3.7}$$

where A, B, and C are three non-collinear points in the net.

$$\begin{aligned} n_y=\sin(-\alpha)\,\|\vec{n}\| \quad &\Leftrightarrow \quad \alpha=-\arcsin\left(\frac{n_y}{\|\vec{n}\|}\right) \\ \frac{n_x}{n_z}=\tan(\beta) \quad &\Leftrightarrow \quad \beta=\arctan\left(\frac{n_x}{n_z}\right) \end{aligned} \tag{3.8}$$

After these estimations, a simulation was run in MATLAB for an interval of $[-80°, 80°]$ for both pitch and yaw and the error was calculated by comparing the result of the estimation with the actual values of both angles. The results can be seen in Figure 3.4. The maximum values for these errors were:

$$\begin{aligned} \max(\varepsilon_\alpha) = &\quad \varepsilon_\alpha(\alpha=40°, &\beta=-80°) &= 21.75°, \\ \min(\varepsilon_\alpha) = &\quad \varepsilon_\alpha(\alpha=-32°, &\beta=-80°) &= -11.40°, \\ \max(\varepsilon_\beta) = &\quad \varepsilon_\beta(\alpha=6°, &\beta=38°) &= 10.49°, \\ \min(\varepsilon_\beta) = &\quad \varepsilon_\beta(\alpha=6°, &\beta=-40°) &= -13.27° \end{aligned}$$

One thing that can be noted is that the expected operating range is smaller since the AUV can know the predicted location of the net and position itself accordingly. Because of this, variation in the angles will be mostly caused by waves, tides, and small displacements of the cage or net.

From the results it can be noticed that the error had positive values when the angles were positive, and negative when the angles were negative. It can also be seen that, in the $[-45°, 45°]$

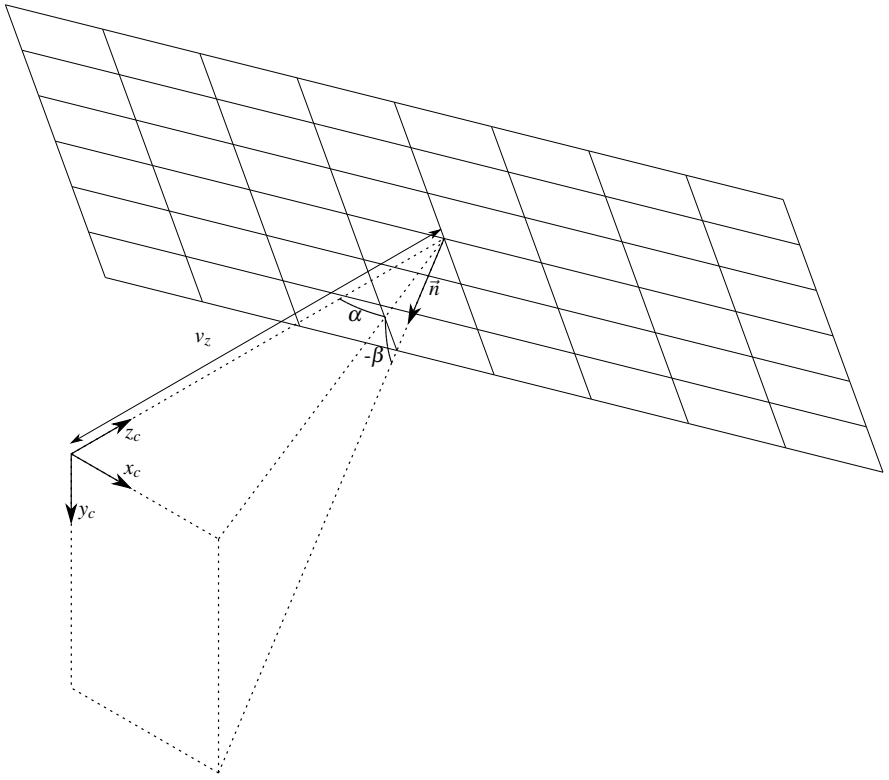
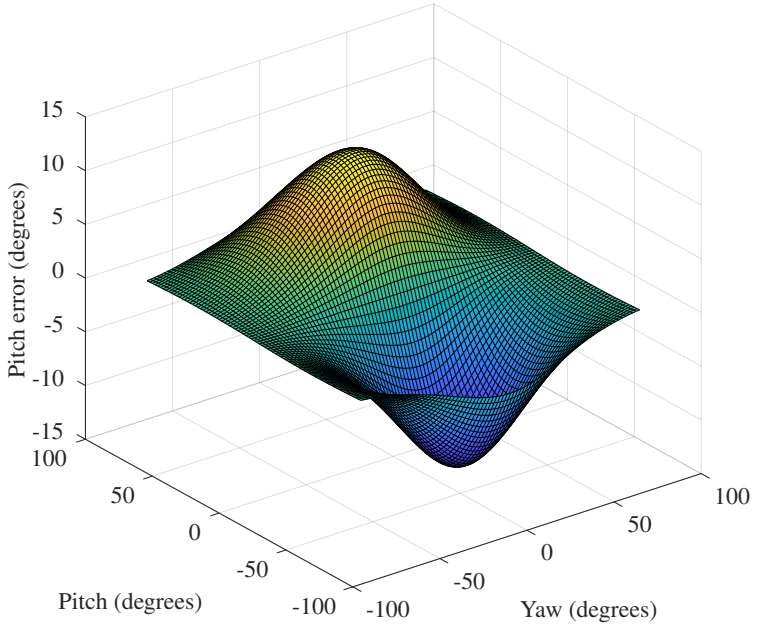
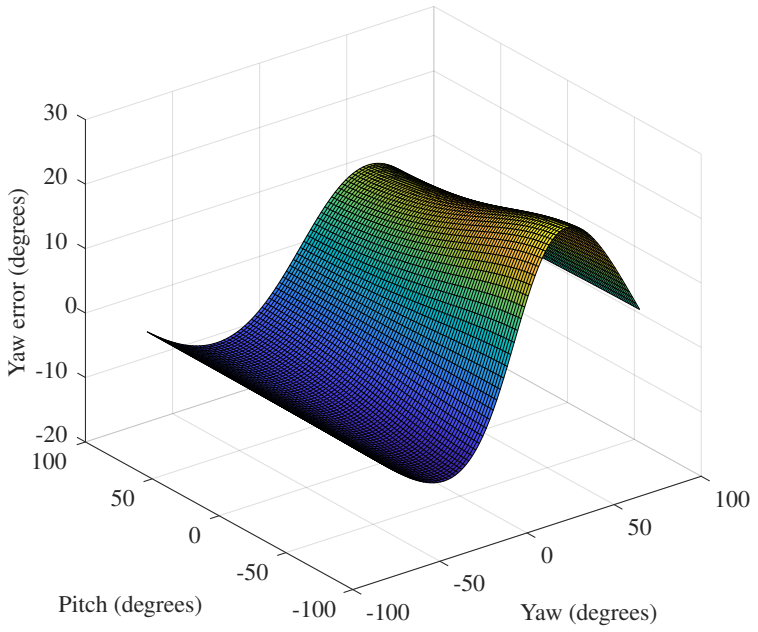


Figure 3.3: Yaw (α) and pitch (β) in relation to the net

range, the error can be approximated by a plane. Because of this, if only this range is considered, the error can be attenuated by multiplying the estimation by a constant coefficient.



(a) Pitch Error



(b) Yaw Error

Figure 3.4: Error in angle estimation

Chapter 4

Implementation

The implementation consisted in the processing of a video to obtain information about the camera's pitch and yaw angles and the distance relative to the camera center.

In order to test the algorithm it is important to know the real orientation of the camera as well as the distance from the center of the image. Due to the difficulty of obtaining this information when dealing with real underwater scenarios, an application was developed in order to synthesize an image similar to one that could be obtained from a camera in a real environment.

These images were then pre-processed in order to extract the information necessary for the implementation of the algorithm described in Chapter 3. Finally, the pitch, yaw, and distance were calculated and displayed. The results of the proposed algorithm were compared with the expected results to assess the validity of the algorithm.

4.1 Image Acquisition

Image acquisition started with obtaining test images from a fresh water tank with a white nylon net and the BlueROV2 camera (Figure 4.1). These images were used as a baseline for a generator of test data to provide greater control over the test environment.

4.1.1 Test Image Generation

To overcome the difficulty of recording underwater images and, more importantly, to know the true values of the data the algorithm needs to extract, an application was developed to synthesize images based on those values (yaw, pitch, and distance). This application considers a set of realistic assumptions that simplify the analysis of the results created by the algorithm.

The camera's intrinsic matrix (2.3) can be obtained through calibration before deployment. Because of that, some information was shared between the image generation and image processing.

In order to simplify the process, the assumptions about the video that were made were:

1. Lens distortion can be removed through calibration.

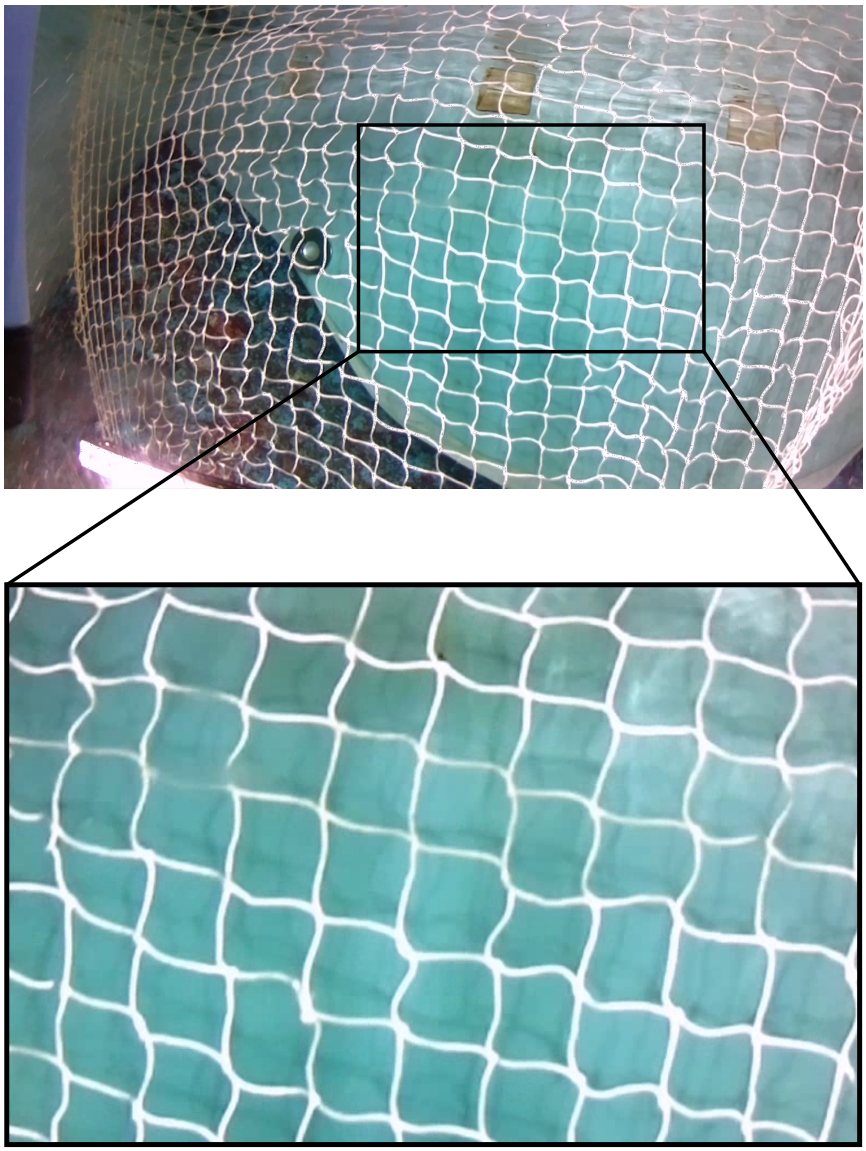


Figure 4.1: Original image obtained from BlueROV2

2. The skew parameter of the intrinsic matrix, s, is considered to be zero since this is the case for most commercial cameras.

3. Even lighting can be achieved either from pre-processing or from a light source in the vehicle.

4. The vehicle is stable in the roll component.

To configure the video generation some variables had to be set. Some of these were passed on to the video processing component, specifically the ones which can be obtained through calibration.

1. Dimensions of the net loops.

2. Camera field of view.
3. Initial angles and translation of the net with respect to the camera's axis.
4. Rate of translation and angle change.
5. Sinusoidal component in angle change.

In order to generate the video for testing, a mesh was generated in the three dimensional space and its 2D representation was calculated through the use of the projective camera matrix (2.3). The homogeneous coordinate system was used to simplify the calculation of the projective 2D representation.

Perlin noise was generated in order to provide a noisy background for the video. The points' 2D representation were connected by a line in order to generate a representation of the net. This net and the background noise were added in order to produce the final video.

An example frame of the video generated can be seen in Figure 4.2. This figure represents the image seen by the vehicle with a 30° yaw angle and 0° pitch angle.

Despite the fact that the generated images do not suffer from the artifacts naturally present in a real situation, this approach has been very useful in order to validate the technique implemented and described in this chapter.

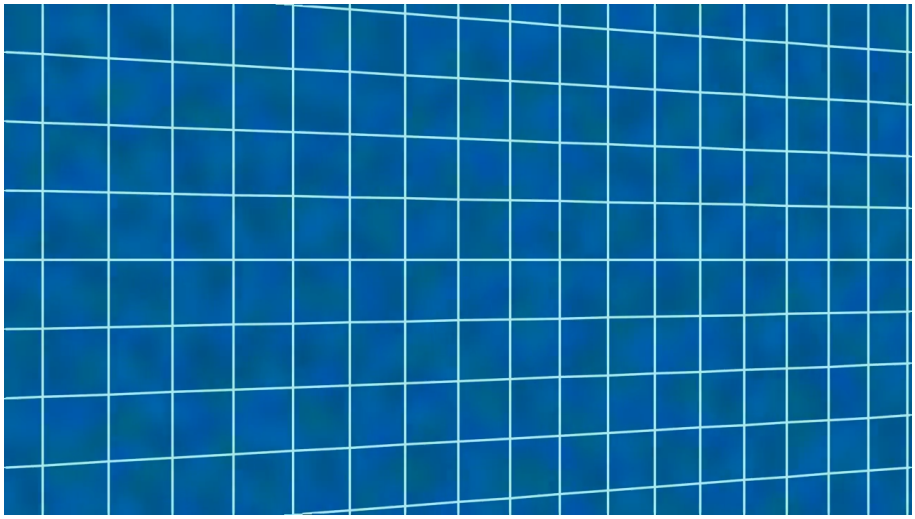


Figure 4.2: Frame of an example video

4.2 Image Processing

After the image was obtained the aim was to extract the image of the net from the background. For this purpose pre-processing was done to obtain the background which then was extracted from the original image. From the resulting image, thresholding was applied in order to obtain a binary image of the net. From this image the contours were found and their characteristics were used in further processing.

The tool used for pre-processing was *OpenCV* [9] and the language was C++.

4.2.1 Pre-Processing

Since one of the assumptions made was that the image had no lens distortion, the filtering done was mainly to separate the net from the background. For this purpose two filters were implemented, one to extract the background, and a second one to remove noise in the image.

The first step was to resize the image to a lower resolution in order to diminish the processing overhead caused by the filtering process. This was done using linear interpolation to calculate the pixel values.

The filters used were median and Gaussian (Section 2.4.2.3). These were chosen because the net is a thin line in an almost uniform background. The Gaussian filter kernel was obtained using the OpenCV function *getGaussianKernel* which uses the following standard deviation σ calculation:

$$\sigma = 0.3\,(\,0.5\,(k-1)-1\,)+0.8\,, \tag{4.1}$$

where k is the kernel size.

In order to extract the background, a median filter with a large kernel was used and the result was filtered again with a Gaussian filter. The other image was smoothed by a Gaussian filter which was followed by a small median filter.

The values which provided the best results were obtained through trial and error using the video generated. Because of that, the specific values used might not work in another scenario. Three filters were applied in the following order: Gaussian, median, and Gaussian. In Table 4.1 the values presented are the kernel sizes used.

Table 4.1: Kernel Sizes for pre-processing filters.

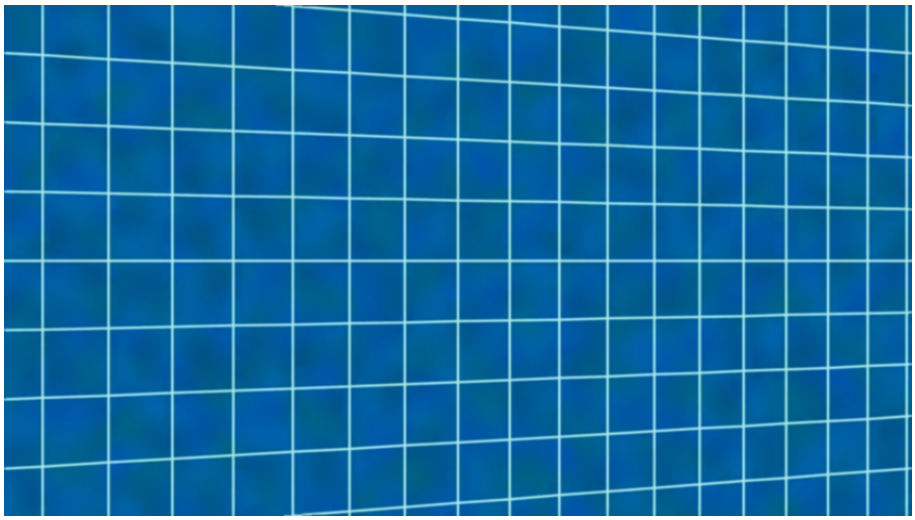
	Kernel Size k	
	Background	Smoothing
First Gaussian	3	31
Median	39	7
Second Gaussian	23	3

The results of the filtering can be seen in Figure 4.3. These were obtained by filtering Figure 4.2 with the previously mentioned filters. The same filters applied to the original cropped image (Figure 4.1) can be seen in Figure 4.4

4.2.2 Image Segmentation

Segmentation of the image was done in order to extract the net as a binary image. The images that are used as input consist of the extracted background and a filtered input. Each image has three

(a) Background extraction



(b) Input smoothing

Figure 4.3: Results of image smoothing

channels, one for each color: red, green and blue.

Initially the background image was subtracted from the smoothed input, by subtracting the value of each pixel in the background image from the corresponding pixel in the in the smoothed image. This operation is done for each of the channels. The result is a colored image where the highest intensity values represent the pixels that are the most different from the background i.e. the net.

Afterwards, a grayscale conversion was done, this is a simple weighted sum:

$$\mathrm{Y} = 0.299\,\mathrm{R} + 0.587\,\mathrm{G} + 0.114\,\mathrm{B}\,, \tag{4.2}$$

where Y represents the final intensity value, and R, G, and B represent the intensities of the colors red, green, and blue, in each pixel, respectively.

Finally, a threshold operation was done (Section 2.4.2.1). The type of threshold used was a binary threshold with a value of 30, which means that every value above it, in a possible range of $0-255$ (8-bit representation of pixel intensity values), was considered part of the net. The value of the threshold was again chosen through trial and error and a different value might be more suitable in other images.

The output is a binary image where ones, in white, represent the net and zeros, in black, represent the background (Figure 4.5).

4.2.3 Data Extraction

After the net was identified and binarized, the next step is to extract the relevant information in order to calculate the pitch and yaw angles. This information is the area and centroid or center of mass of the net loops.

The binary image resultant of the previous process had well defined boundaries around the loops so no further processing was required. However, when applying the same filters to the net in Figure 4.1, some gaps can be seen in the net lines (Figure 4.6a). A morphology closing operation can be applied to improve these images and bridge the gaps. This can observed in Figure 4.6b, where a closing operation with a disk of radius 10 was used. Figure 4.6 has marked, in blue, the successful gap bridging mentioned before, and, in red, undesired artifacts that also happen as a side effect of this operation.

After, the contours of the regions of the image are extracted using the *OpenCV* function *findContours* which returns, in this situation, a representation of the edges of the loops of the net. This was used over edge detection since it ensures that the the edge consists of a closed loop and returns a separate element for each of these closed loops.

The area and centroid were calculated for every contour. The area was used as a filtering argument in order to properly select the contours that in fact represent a loop in the net.

4.3 Contour Matching

Until now, the processing has been done as if each frame of the video was a individual image. The truth is that the image is part of a sequence that represents the movement of a net. What this means is that it is useful to match each loop with its correspondent contour.

In order to match each contour with a loop, the first step is to assign an identification number to every loop in the initial frame. Then, match the contours of the current frame with the ones in the previous in order to maintain their initial identification number.

The matching was done by calculating a matrix which represents the distances between the centroids of the contours in the previous frame and the ones in the current frame. Then, finding a match by calculating the minimum values of this matrix. This can be done by assuming that the movement of the net is smaller than one loop size between frames, to not incur in aliasing.

These minimums were chosen using a greedy algorithm represented in Algorithm 4.1. The algorithm starts by finding the minimum value of each column and each row and storing them.

Then, for every row or column, depending on the sizes of the input (the biggest is chosen), it checks if the value is the minimum in both its row and column. Then, for every value where this is true, it stores the value and marks both its column and its row to be ignored in the next iteration. This process is repeated until an element has been chosen for every row of column, depending on the sizes of the input (the smallest one is chosen).

```
1  while ! all matches found do
2      foreach row not ignored do
3          find minimum value
4      end foreach
5      foreach column not ignored do
6          find minimum value
7      end foreach
8      foreach minimum in row or column do
9          if element is minimum in both row and column then
10             matches ← element
11             ignored rows ← element row
12             ignored columns ← element column
13         end if
14     end foreach
15 end while
```

Algorithm 4.1: Contour matching

4.4 Heading Estimation

After getting the information about the areas of the net loops and their centroids, the next step was to estimate the heading of the camera with respect to the net. To do this, a plane was fit to the 3D conversion of the points in the image. Afterwards, the heading was calculated using the angles that the plane normal formed with the camera axis.

4.4.1 Plane Estimation

The first step was to convert the points into their 3D estimation. The distance from area was calculated using (3.6). The x and y components were calculated using:

$$x_{3D} = \left(u - \frac{w}{2}\right)\frac{1}{\sqrt{A}}, \quad y_{3D} = \left(v - \frac{h}{2}\right)\frac{1}{\sqrt{A}} \quad ,$$

where u and v are the pixel coordinates, w and h are the image dimensions height and width, and A is the area of the contour.

Plane estimation was done using Random Sample Consensus (RANSAC). The approach used can be seen in Algorithm 4.2.

The objective of this algorithm is to find the best estimation for the plane. After variable initialization, it chooses three points at random positions and calculated the plane using (3.7). Then, it goes through all the points, except for those that generated the plane, and calculates the distance between them and the plane. In the case that the distance is smaller than a set distance, the point is added to the list of points that validate the plane generated. At the end, if the number of matches is high enough, it validates the model and checks whether the current model is better than the previous ones (this is done by calculating the average distance of the matches to the plane). This process is done for a pre-determined number of times and the best model is selected.

```
1  iterations = 0
2  best fit = Null
3  minimum error = large number
4  while iterations < k do
5      random points = 3 random points from the set
6      model = plane generated by random points
7      model matches = empty set
8      total distance = 0
9      foreach point in set not in random points do
10         if distance to plane < minimum distance for match then
11             add point to model matches
12             total distance += distance to plane
13         end if
14     end foreach
15     if number of points in model matches > d then
16         average distance = (total distance) / (model matches)
17         if average distance < minimum Error then
18             best fit = model
19             minimum error = average distance
20         end if
21     end if
22     iterations++
23 end while
```

Algorithm 4.2: RANSAC iterative method

The distance from the plane to a point P was calculated using:

$$d = \frac{\left|\overrightarrow{AP} \cdot \vec{n}\right|}{\|\vec{n}\|}, \tag{4.3}$$

where $\overrightarrow{AP}$ is the vector that goes from the point A in the plane to the point P, and $\vec{n}$ is the normal vector to the plane.

RANSAC is used to calculate a model excluding outliers that might exist. This method was used because small variations in the areas obtained can make a big difference in the distance estimation.

4.4.1.1 Pitch, Yaw, and Distance

Pitch and yaw were calculated using the results from the plane estimation (3.8).

Because the information obtained from the image generation was the angles used to rotate the set of points, this was the information used to compare the results of the image processing.

Distance was calculated by obtaining the point in the plane that crosses the axis z_c. The plane equation is given by:

$$ax + by + cz + d = 0, \tag{4.4}$$

where,

$$\begin{aligned} a = |\vec{n}_x|, \quad b = |\vec{n}_y|, \quad c = |\vec{n}_z|, \\ d = -\left(|\vec{n}_x|A_x + |\vec{n}_y|A_y + |\vec{n}_z|A_z\right), \end{aligned} \tag{4.5}$$

where A is a point in the plane and $\vec{(n)}$ is the normal to the plane.

By replacing x and y in the plane equation by 0, the intersection of the plane with the axis z_c can be obtained.

4.4.1.2 Distance Error Correction

In the initial distance estimation the angular component was removed which introduced an error in the distance estimation. To compensate for this, since the angular component has now been estimated, the yaw and pitch angle were used to correct the original estimate.

If (3.5) is used, and the angles are replaced, the result is a denominator with a fourth order polynomial function which does not have a trivial solution.

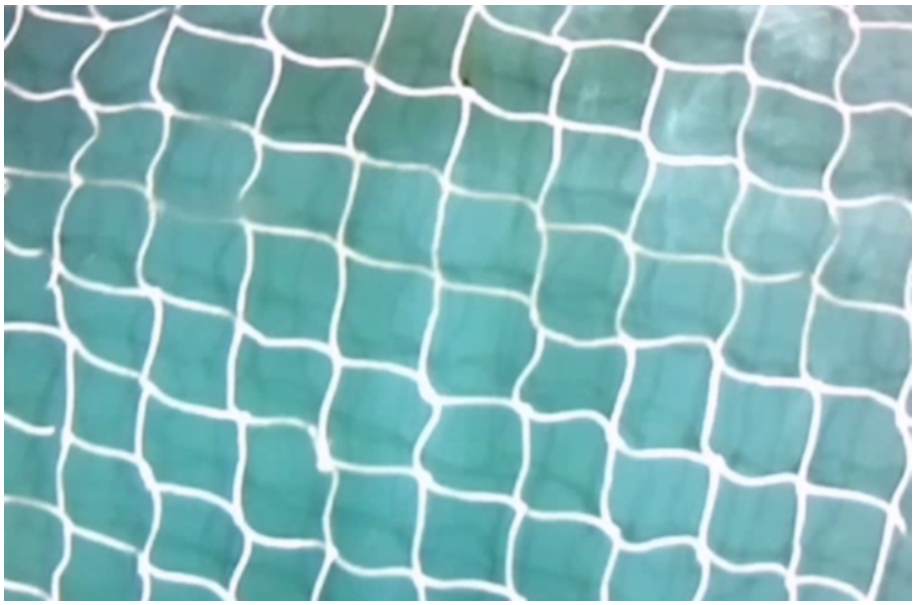
It can be seen that the influence of the angles in the denominator are much smaller than in the numerator. This is due to the fact that the sin() and cos() functions have a maximum amplitude of 1. Because of this, the approximation that can be made is to remove once again the angular influence, this time only from the denominator. This results in a much simpler equation:

$$A = \frac{l^2 \left|\cos(\alpha)\cos(\beta)\right|}{\tan\left(\frac{\theta}{2}\right)^2 v_z^2} \tag{4.6}$$

From here, a more accurate distance can be approximated by:

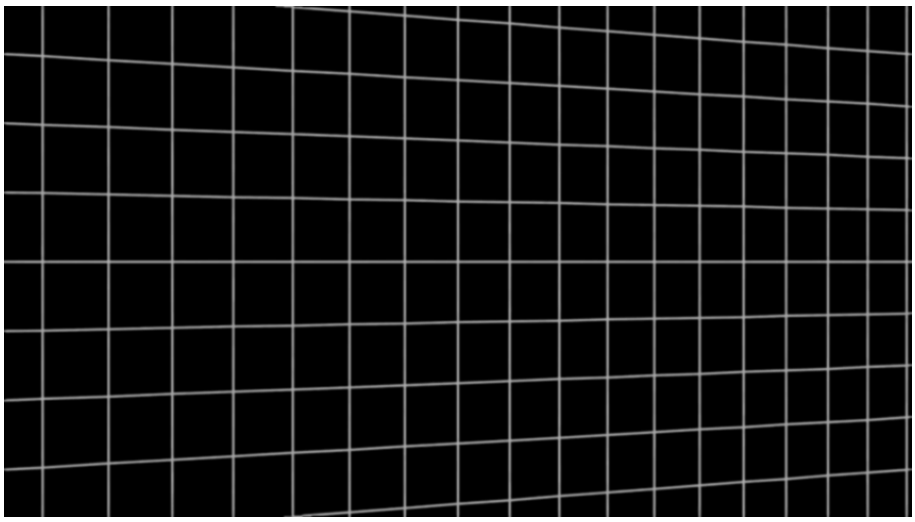
$$v_z \approx \frac{l\sqrt{\left|\cos(\alpha)\right|\left|\cos(\beta)\right|}}{\tan\left(\frac{\theta}{2}\right)\sqrt{A}} \tag{4.7}$$

(a) Background extraction

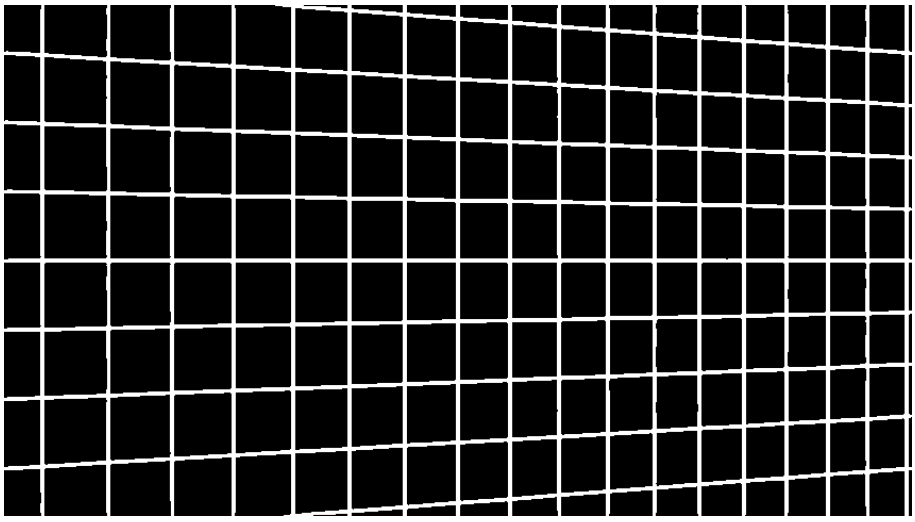


(b) Input smoothing

Figure 4.4: Results of image smoothing on the cropped frame

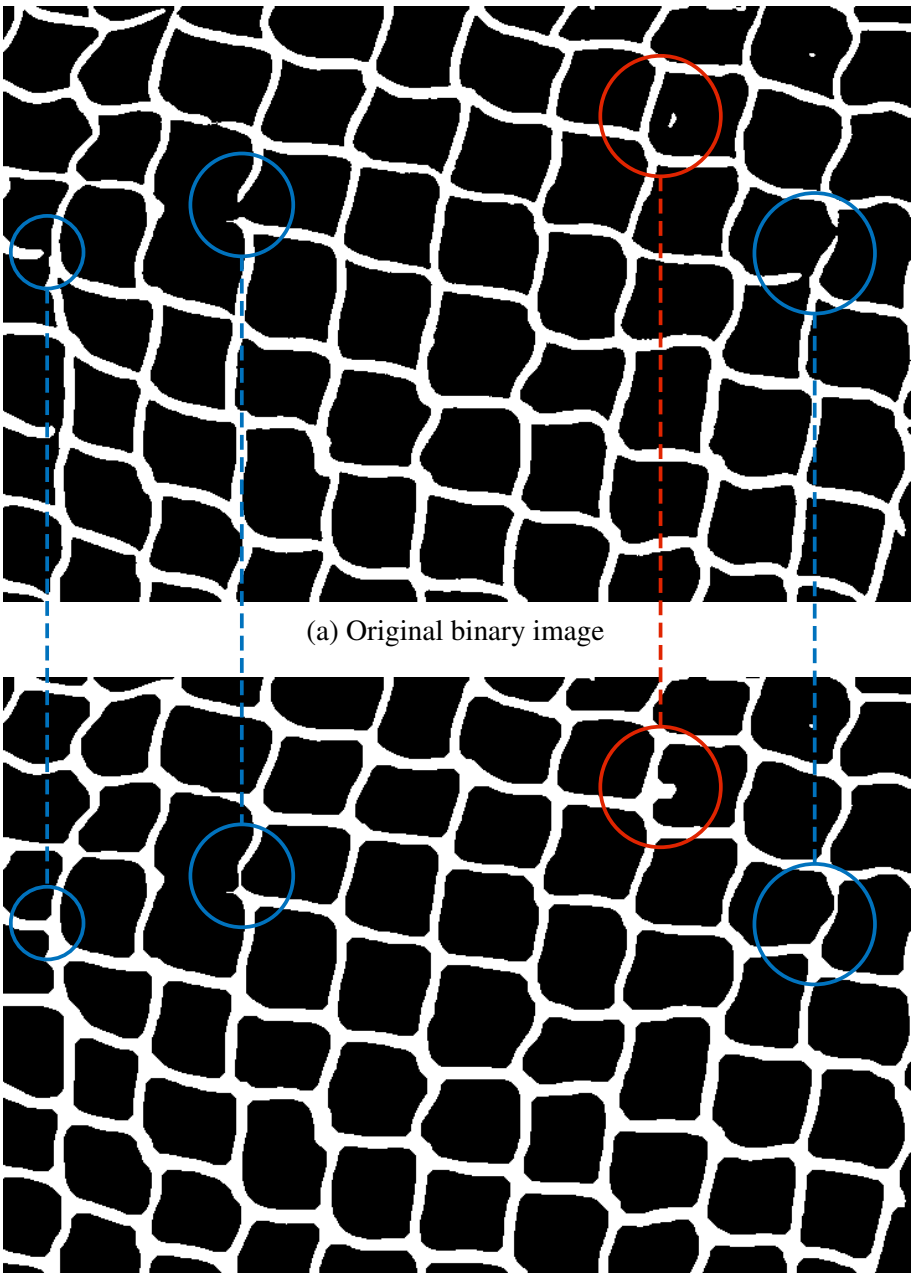


(a) Grayscale result of subtraction



(b) Binary Output

Figure 4.5: Net extraction



(a) Original binary image

(b) Binary image after closing

Figure 4.6: Results of closing operation

Chapter 5

Results

Here are presented the results obtained by the proposed algorithm, for a selected set of test images. Three videos were selected that best exemplify the overall results obtained. The data treatment was done after obtaining the raw data from the system because MATLAB allowed for a more immediate visualization of small changes in post-processing.

The values for the distance estimation are normalized to the size of the net loops, as there is a linear relation between that dimension and the distance between the camera and the net. Because of this, the distance measurements will be referenced as 'unit'.

5.1 Post-Processing

Post-processing was done in order to improve the raw output of the system and was applied to the three components of the computed data: yaw, pitch, and distance. This was done in two steps, first, a low-pass filter in order to lower noise present in the output, and secondly, error compensation. At first, a simple running average was applied to the three outputs which allowed for a smoother response. This filter considers the current value as the average of the previous values up to a pre determined distance. The first few values of the filtered output were ignored because MATLAB considers the values before the data as 0.

Secondly, the pitch and yaw were multiplied by 1.1. This value was obtained through trial and error and provided the best results in the range of videos used for testing.

To improve the distance estimation the pitch and yaw were used. From (3.6) and (4.7) the equations are:

$$\begin{cases} d_{old} = \dfrac{l}{\tan\left(\frac{\theta}{2}\right)\sqrt{A}} \\ d_{new} = \dfrac{l\sqrt{|\cos(\alpha)||\cos(\beta)|}}{\tan\left(\frac{\theta}{2}\right)\sqrt{A}} \end{cases} \Rightarrow d_{new} = d_{old}\sqrt{|\cos(\alpha)||\cos(\beta)|}\,, \tag{5.1}$$

where d_{new} is the improved approximation and d_{old} is the previous estimation.

5.2 Implementation Results

The videos used had 400 frames each and the parameters used for their generation can be seen in Table 5.1. The first video was chosen to show the system response to a linear angle variation, in the system's expected range of operation.

The second video was chosen to display the response to a variation of both angle and distance.

Finally, the third video was chosen to display the system response to a more complex angle variation, where both angles vary according to sinusoidal functions.

Table 5.1: Video parameters

	Pitch Angle			
	Linear		Sinusoidal	
	Initial (deg)	Rate (deg/frame)	Amplitude (deg)	Frequency (1/video)
Video 1	0	0	0	0
Video 2	0	0	0	0
Video 3	0	0	30	3

Table 5.1: Video parameters (cont.)

	Yaw Angle				Distance	
	Linear		Sinusoidal		Linear	
	Initial (deg)	Rate (deg/frame)	Amplitude (deg)	Frequency (1/video)	Initial (unit)	Rate (unit/frame)
Video 1	-60	0.30	0	0	30	0.00
Video 2	-15	0.10	10	3	30	-0.01
Video 3	0	0.00	45	2	40	0.00

5.2.1 First Video

In Figure 5.1 it is possible to see that the error in pitch estimation does not change significantly with the change in yaw. It can also be noticed that after the simple moving average filtering that was used, the error was significantly reduced.

In Figure 5.2, it can be noticed that the shape of the error is consistent with the expected from the calculations made in Section 3.4. Despite the shape being similar, the amplitude of the error is much smaller possibly due to the fact that the area used in the calculation was different from the one obtained from the video.

Finally, in Figure 5.3 it can be noticed the influence of the angle in the distance estimation and the difference the error compensation made, since the initial errors were above 10 and the final error was below 1.3.

5.2.2 Second Video

In Figure 5.4 and 5.5 it can be seen that the distance variation does not significantly influence the angle estimation.

Secondly, in Figure 5.9 it can be noticed that the error remains constant despite the variation in distance coupled with the variation in angle.

5.2.3 Third Video

In Figure 5.7 and Figure 5.8 it can be noticed that the estimation of each angle is not greatly influenced by the variation of the other.

In Figure 5.9 as in the first video, it can be noticed the influence of the angles in the distance estimation and, like before, the influence the error correction had in the final result.

5.2.4 Summary of the Results

The maximum error for each video after post-processing can be seen in Table 5.2. These are acceptable results for the application considered, which is the positioning of an AUV relative to a slow moving net for inspection purposes.

Table 5.2: Maximum absolute error

	Pitch Angle (deg)	Yaw Angle (deg)	Distance (unit)
Video 1	1.69	3.00	1.10
Video 2	1.36	3.31	1.22
Video 3	2.57	2.13	1.07

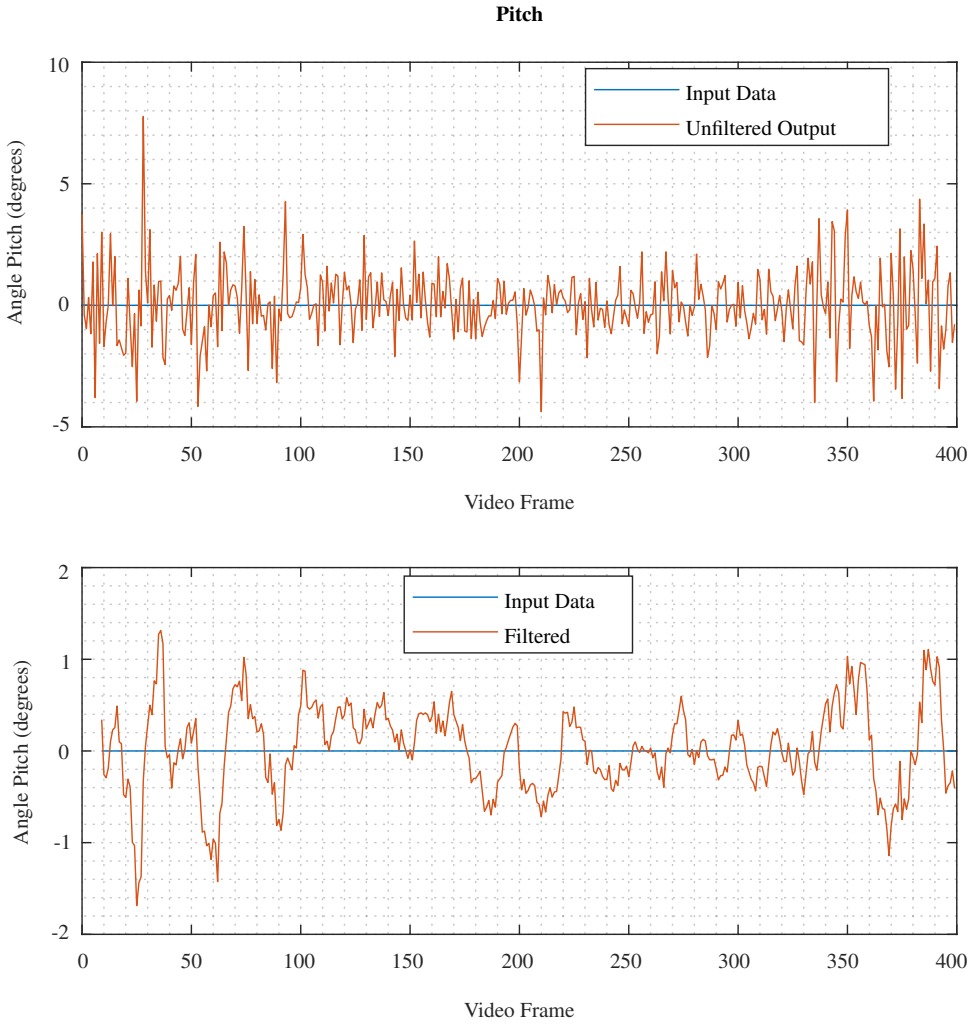


Figure 5.1: Video 1 Pitch Error. Video settings were: pitch= 0, linear variation in yaw, and constant distance

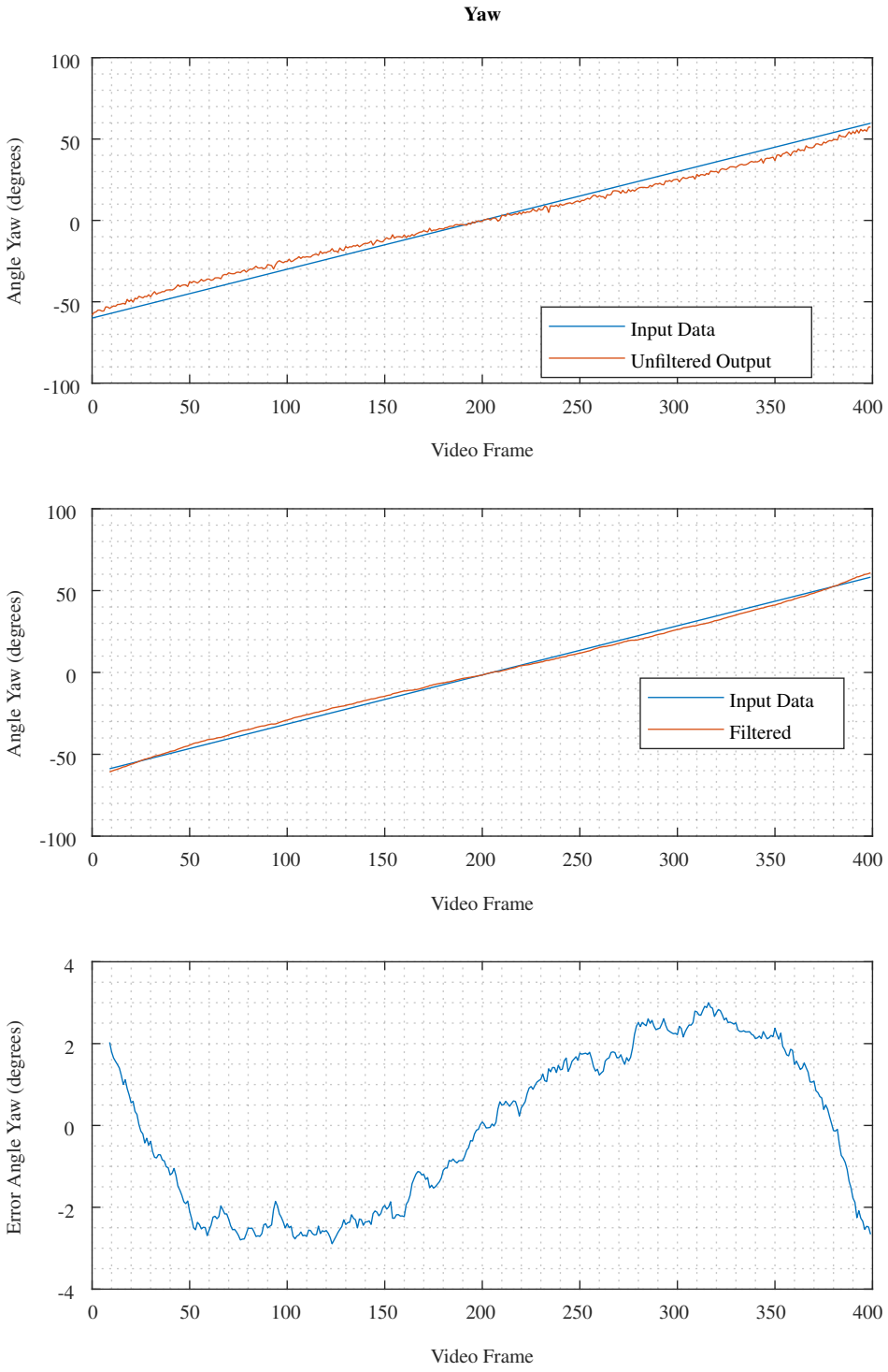


Figure 5.2: Video 1 Yaw Error. Video settings were: pitch= 0, linear variation in yaw, and constant distance

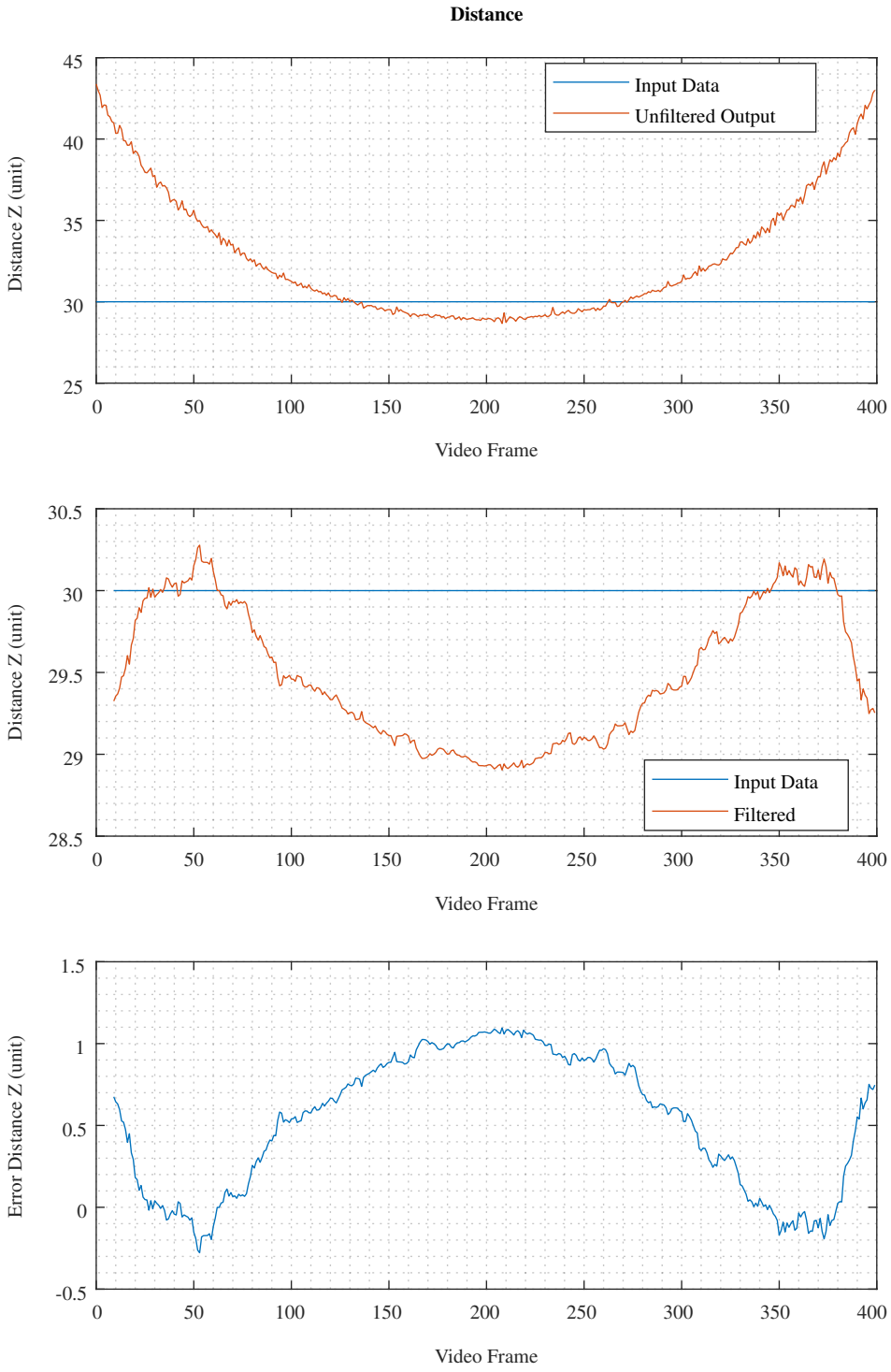


Figure 5.3: Video 1 Distance Error. Video settings were: pitch= 0, linear variation in yaw, and constant distance

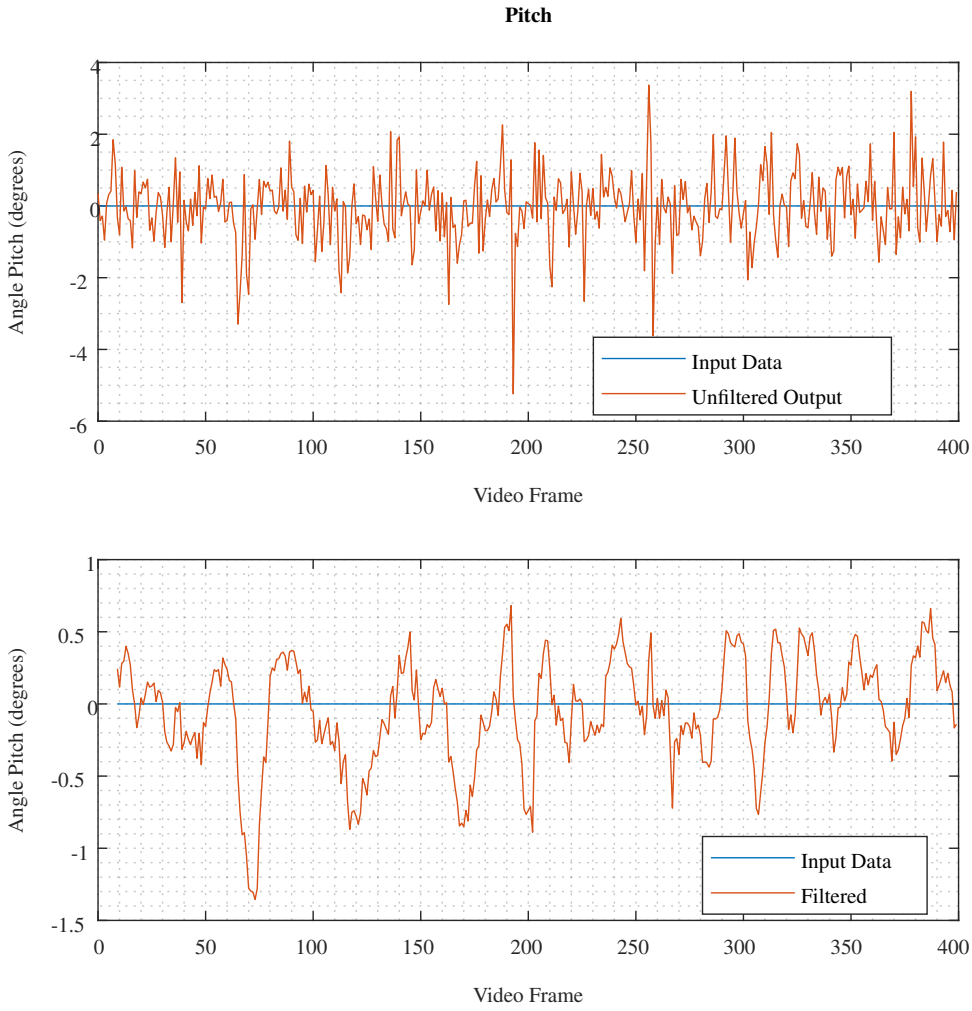


Figure 5.4: Video 2 Pitch Error. Video settings were: pitch= 0, linear and sinusoidal variation in yaw, and linear variation in distance

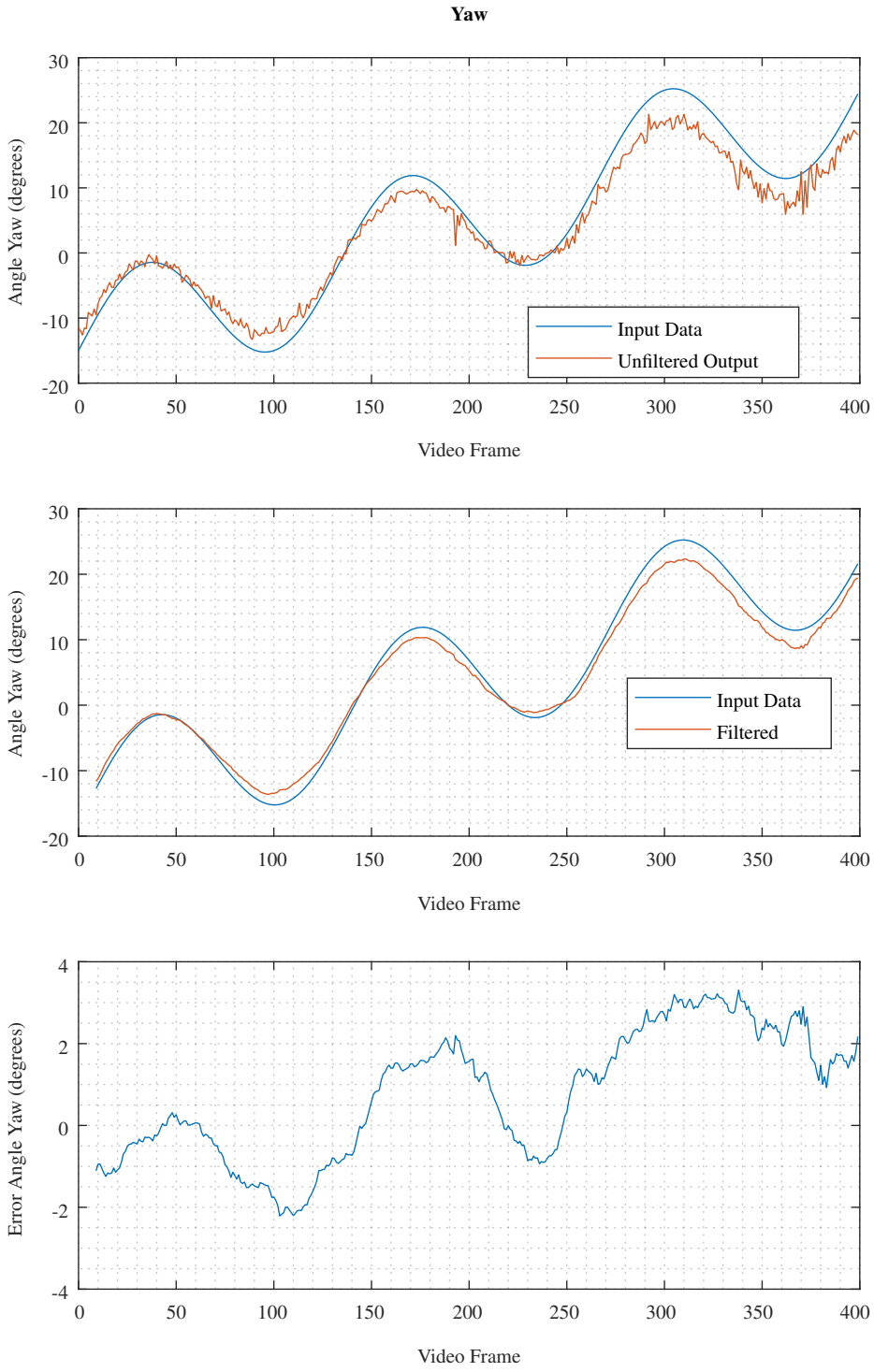


Figure 5.5: Video 2 Yaw Error. Video settings were: pitch= 0, linear and sinusoidal variation in yaw, and linear variation in distance

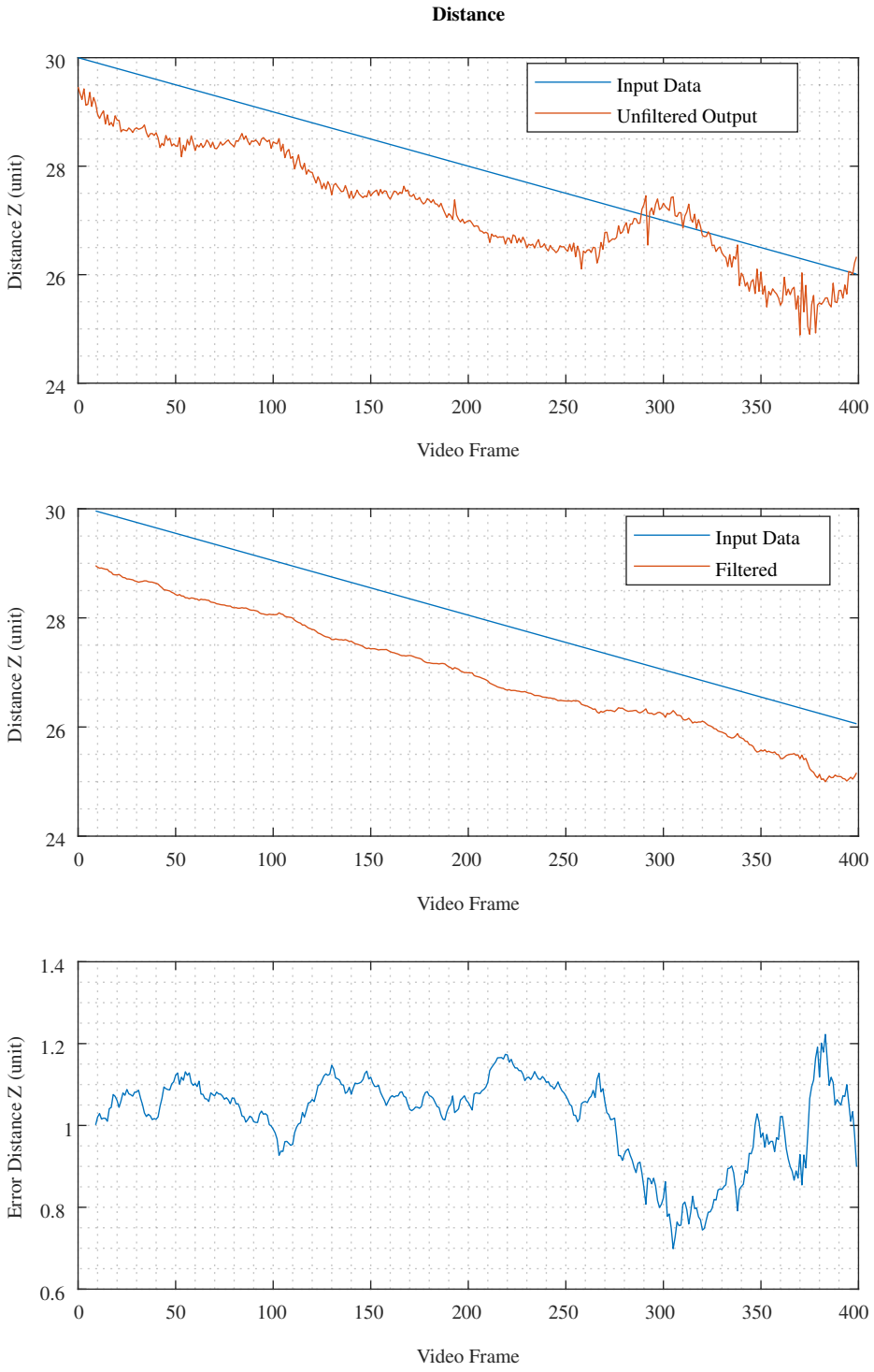


Figure 5.6: Video 2 Distance Error. Video settings were: pitch= 0, linear and sinusoidal variation in yaw, and linear variation in distance

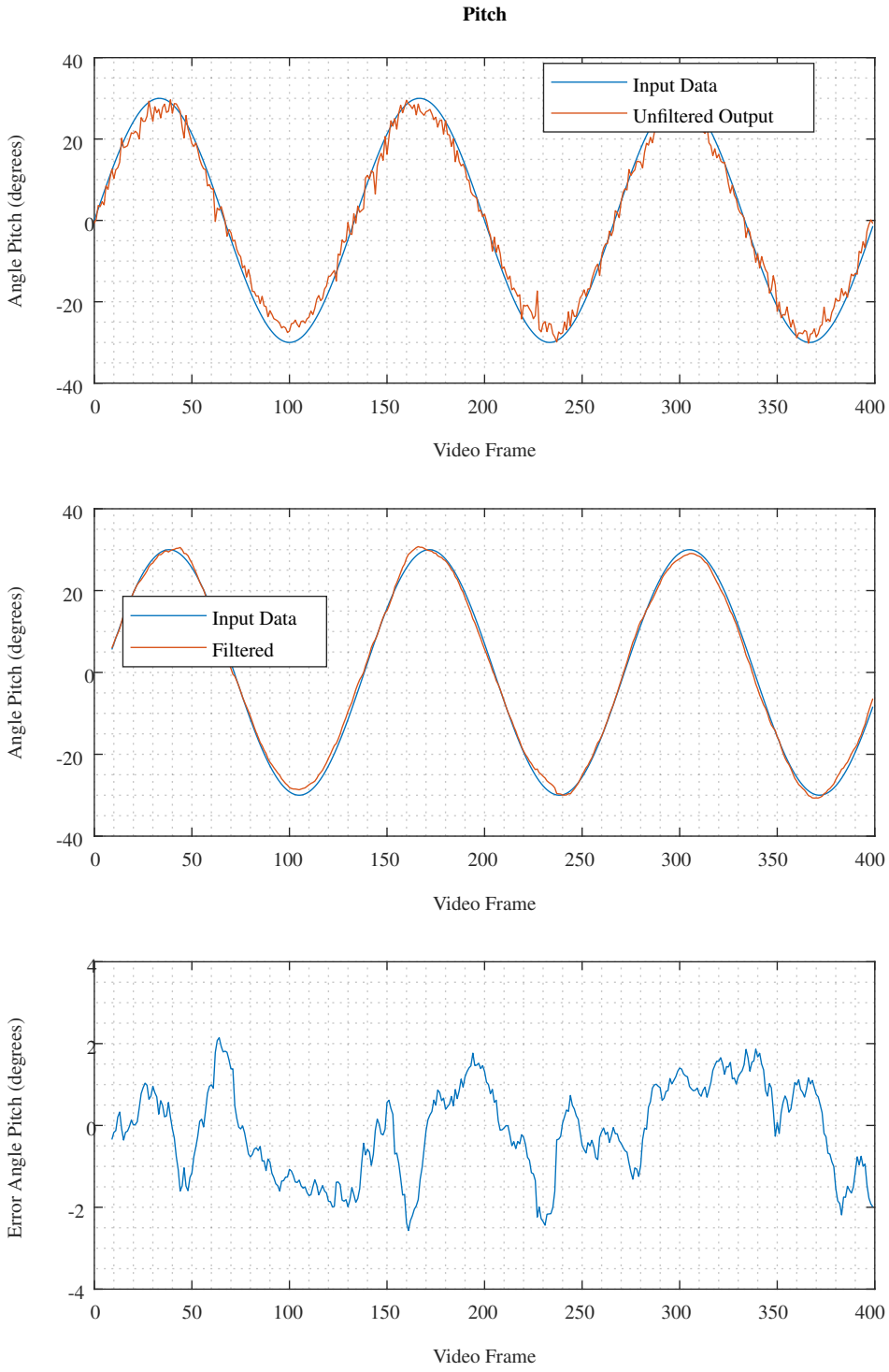


Figure 5.7: Video 3 Pitch Error. Video settings were: sinusoidal variation in pitch, sinusoidal variation in yaw, and constant distance

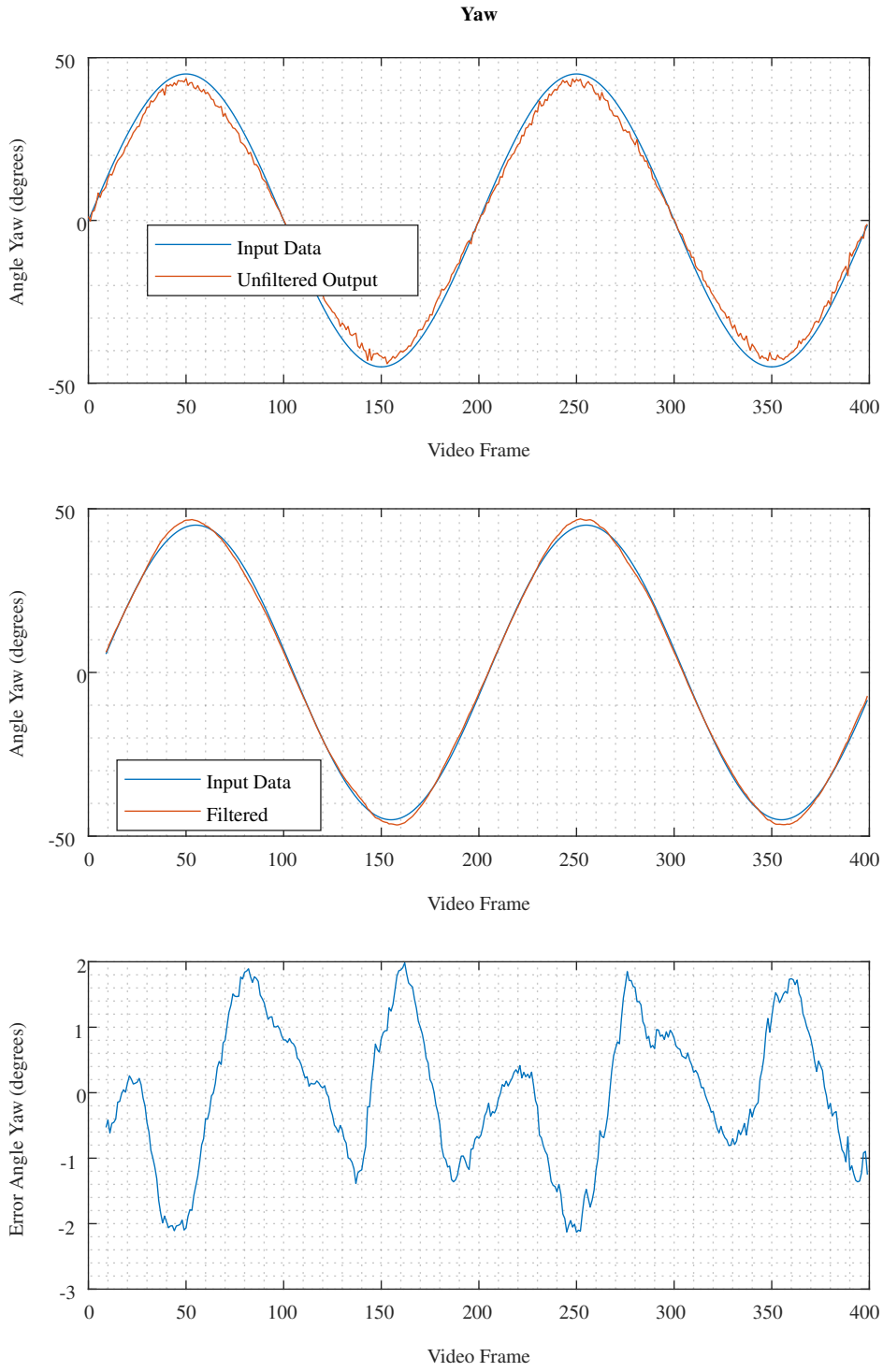


Figure 5.8: Video 3 Yaw Error. Video settings were: sinusoidal variation in pitch, sinusoidal variation in yaw, and constant distance

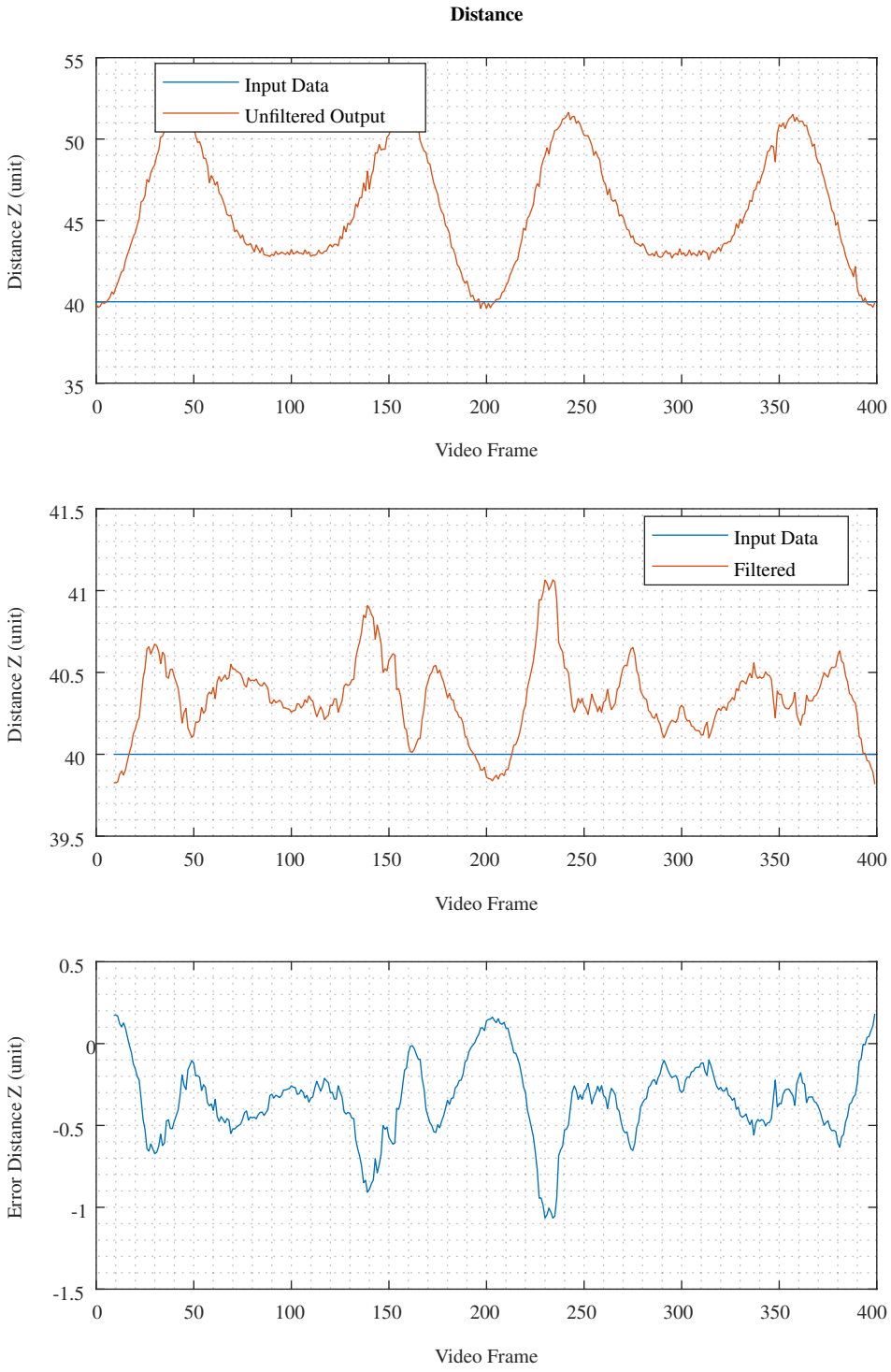


Figure 5.9: Video 3 Distance Error. Video settings were: sinusoidal variation in pitch, sinusoidal variation in yaw, and constant distance

Chapter 6

Conclusions

The objective of this dissertation was to develop a system for positioning of an AUV relative to an aquaculture net or cage using image processing with a single camera. To do this, an approach based on the relation between distance, perspective, and the known dimension of the loops of the net was taken.

At first, an analysis of the perspective camera projection was made, and the relation between the distance and area of a theoretical net loop was studied. It was shown that this relation is dependent on the angle relative to the camera.

Because of this dependence, an approach was taken to approximate the distance of each loop by removing the angular component. This approach introduced an error which was later compensated.

The angle relative to the net was calculated based on the relation between the distances of the loops relative to the camera. These estimations were then used to diminish the error that was introduced when the initial distance was calculated.

To test the approach, a synthetic video was created which approximated the physical situation that was obtained in a controlled environment.

Finally, the results of the method were shown. These results show the possibility of the usage of the method even if some improvements need to be made in order to further reduce the error.

6.1 Future Work

Several improvements can be made in order to better approximate the obtained data to the real world.

First, better data can be generated in order further approximate the input data to a real situation. In order to do this, data from a real aquaculture net or cage could be used.

The real life scenario lacked the artifacts that occur in real underwater video. Because of that, the pre processing step of the approach is also missing the necessary elements that remove these artifacts.

An improved matching algorithm can be used, one that accounts for the fact that a net loop might not be recognized for a few frames. A better tracking algorithm can also be used in order to add a horizontal and vertical movement calculation.

A better filter can be used in the post processing step to further reduce the error. For a better filter, a better knowledge of the movement that a net is subject to, together with other sensory data can be used.

References

[1] B. Allotta, A. Caiti, R. Costanzi, F. Fanelli, D. Fenucci, E. Meli, and A. Ridolfi. A new AUV navigation system exploiting unscented Kalman filter. *Ocean Engineering*, 113:121–132, February 2016. http://linkinghub.elsevier.com/retrieve/pii/S0029801815007271.

[2] Vaibhav G. Awale and Hari B. Hablani. Fusion of Navigation Solutions from Different Navigation Systems for an Autonomous Underwater Vehicle. *IFAC Proceedings Volumes*, 47(1):635–642, 2014. http://linkinghub.elsevier.com/retrieve/pii/S1474667016327227.

[3] Graeme J. Awcock and Ray Thomas. *Applied Image Processing*. McGraw-Hill, Inc., 1995.

[4] Alexander Bahr, John J. Leonard, and Alcherio Martinoli. Dynamic positioning of beacon vehicles for cooperative underwater navigation. pages 3760–3767. IEEE, October 2012. http://ieeexplore.ieee.org/document/6386168/.

[5] B. Barshan and H.F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, June 1995. http://ieeexplore.ieee.org/document/388775/.

[6] Brian Bingham, Brendan Foley, Hanumant Singh, Richard Camilli, Katerina Delaporta, Ryan Eustice, Angelos Mallios, David Mindell, Christopher Roman, and Dimitris Sakellariou. Robotic tools for deep water archaeology: Surveying an ancient shipwreck with an autonomous underwater vehicle. *Journal of Field Robotics*, 27(6):702–717, November 2010. http://doi.wiley.com/10.1002/rob.20350.

[7] Johann Borenstein, Hobart R. Everett, Liqiang Feng, and David Wehe. Mobile robot positioning-sensors and techniques. Technical report, NAVAL COMMAND CONTROL AND OCEAN SURVEILLANCE CENTER RDT AND E DIV SAN DIEGO CA, 1997.

[8] Nathaniel Bowditch. *The American Practical Navigator: An Epitome Of Navigation (Bicentennial Edition)*. 2002.

[9] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[10] John Canny. A computational approach to edge detection. In *Readings in Computer Vision*, pages 184–203. Elsevier, 1987.

[11] Xiang Cao, Hongbing Sun, and Gene Eu Jan. Multi-AUV cooperative target search and tracking in unknown underwater environment. *Ocean Engineering*, 150:1–11, February 2018. http://linkinghub.elsevier.com/retrieve/pii/S0029801817307655.

[12] A Caruso, F Paparella, L F M Vieira, M Erol, and M Gerla. The Meandering Current Mobility Model and its Impact on Underwater Mobile Sensor Networks. page 9, 2008.

[13] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110, June 2006. http://ieeexplore.ieee.org/document/1638022/.

[14] Andre B. Figueiredo, Bruno M. Ferreira, and Anibal C. Matos. Vision-based localization and positioning of an AUV. pages 1–7. IEEE, April 2016. http://ieeexplore.ieee.org/document/7485384/.

[15] A.R. Frost, A.P. McMaster, K.G. Saunders, and S.R. Lee. The development of a remotely operated vehicle (ROV) for aquaculture. *Aquacultural Engineering*, 15(6):461–483, November 1996. http://linkinghub.elsevier.com/retrieve/pii/S0144860996010047.

[16] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003.

[17] Jon Henderson, Oscar Pizarro, Matthew Johnson-Roberson, and Ian Mahon. Mapping Submerged Archaeological Sites using Stereo-Vision Photogrammetry: Mapping submerged sites using Stereo-Vision Photogrammetry. *International Journal of Nautical Archaeology*, 42(2):243–256, September 2013. http://doi.wiley.com/10.1111/1095-9270.12016.

[18] Michael Hoy, Alexey S. Matveev, and Andrey V. Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey. *Robotica*, 33(03):463–497, March 2015. http://www.journals.cambridge.org/abstract_S0263574714000289.

[19] George W. C. Kaye and Thomas H. Laby. The speed and attenuation of sound 2.4.1 Kaye & Laby Online. Version 1.0. In *Tables of Physical & Chemical Constants*. 16th edition, 1995. http://www.kayelaby.npl.co.uk/general_physics/2_4/2_4_1.html.

[20] Jungmin Kim, Yountae Kim, and Sungshin Kim. An accurate localization for mobile robot using extended Kalman filter and sensor fusion. pages 2928–2933. IEEE, June 2008. http://ieeexplore.ieee.org/document/4634210/.

[21] James C. Kinsey, Ryan M. Eustice, and Louis L. Whitcomb. A survey of underwater vehicle navigation: Recent advances and new challenges. In *IFAC Conference of Manoeuvering and Control of Marine Craft*, volume 88, pages 1–12, 2006.

[22] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, June 1991. http://ieeexplore.ieee.org/document/88147/.

[23] John J. Leonard and Alexander Bahr. Autonomous Underwater Vehicle Navigation. In Manhar R. Dhanak and Nikolaos I. Xiros, editors, *Springer Handbook of Ocean Engineering*, pages 341–358. Springer International Publishing, Cham, 2016. https://doi.org/10.1007/978-3-319-16649-0_14.

[24] Alberto Ortiz, Javier Antich, and Gabriel Oliver. A Bayesian approach for tracking undersea narrow telecommunication cables. pages 1–10. IEEE, May 2009. http://ieeexplore.ieee.org/document/5278108/.

[25] Liam Paull, Sajad Saeedi, Mae Seto, and Howard Li. AUV Navigation and Localization: A Review. *IEEE Journal of Oceanic Engineering*, 39(1):131–149, January 2014. http://ieeexplore.ieee.org/document/6678293/.

[26] N.S.V. Rao, S. Kareti, Weimin Shi, and S.S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM–12410, 10180101, July 1993. http://www.osti.gov/servlets/purl/10180101-GX4NXg/.

[27] Ingrid Schjølberg and Ingrid Bouwer Utne. Towards autonomy in ROV operations. *IFAC-PapersOnLine*, 48(2):183–188, 2015. http://linkinghub.elsevier.com/retrieve/pii/S2405896315002694.

[28] Jean Serra. Introduction to mathematical morphology. *Computer vision, graphics, and image processing*, 35(3):283–305, 1986.

[29] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2014.

[30] L. Stutters, Honghai Liu, C. Tiltman, and D.J. Brown. Navigation Technologies for Autonomous Underwater Vehicles. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(4):581–589, July 2008. http://ieeexplore.ieee.org/document/4524846/.

[31] Tomasz Szyrowski, Sanjay K Sharma, Robert Sutton, and Gareth A Kennedy. Developments in subsea power and telecommunication cables detection: Part 1 – Visual and hydroacoustic tracking. *Underwater Technology*, 31(3):123–132, July 2013. http://openurl.ingenta.com/content/xref?genre=article&issn=1756-0543&volume=31&issue=3&spage=123.

[32] Hwee-Pink Tan, Roee Diamant, Winston K.G. Seah, and Marc Waldmeyer. A survey of techniques and challenges in underwater localization. *Ocean Engineering*, 38(14-15):1663–1676, October 2011. http://linkinghub.elsevier.com/retrieve/pii/S0029801811001624.

[33] Juan D. Tardós, José Neira, Paul M. Newman, and John J. Leonard. Robust Mapping and Localization in Indoor Environments Using Sonar Data. *The International Journal of Robotics Research*, 21(4):311–330, April 2002. http://journals.sagepub.com/doi/10.1177/027836402320556340.

[34] Tinne Tuytelaars and Krystian Mikolajczyk. Local Invariant Feature Detectors: A Survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3):177–280, 2007. http://www.nowpublishers.com/article/Details/CGV-017.

[35] Ingrid Utne, Ingrid Schjølberg, and Ingunn Holmen. Reducing risk in aquaculture by implementing autonomous systems and integrated operations. September 2015.

[36] Tao Zhang, Liping Chen, and Yao Li. AUV Underwater Positioning Algorithm Based on Interactive Assistance of SINS and LBL. *Sensors*, 16(1):42, December 2015. http://www.mdpi.com/1424-8220/16/1/42.

[37] Zheping Yan, Shuping Peng, Jiajia Zhou, Jian Xu, and Heming Jia. Research on an improved dead reckoning for AUV navigation. pages 1793–1797. IEEE, May 2010. http://ieeexplore.ieee.org/document/5498571/.

[38] Djemel Ziou and Salvatore Tabbone. Edge detection techniques-an overview. *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, 8:537–559, 1998.